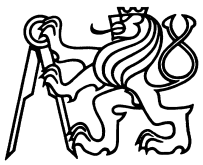
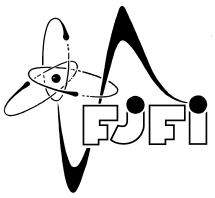


České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská



Variační autoencoder a jeho rozšíření

Variational autoencoder and its extensions

Bakalářská práce

Autor: **Michaela Mašková**

Vedoucí práce: **Doc. Ing. Václav Šmídl, Ph.D.**

Akademický rok: 2019/2020

České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská

Katedra: matematiky

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Michaela Mašková
Studijní program:	Aplikace přírodních věd
Obor:	Matematické inženýrství
Zaměření:	Aplikované matematicko-stochastické metody
Název práce (česky):	Variační autoencoder a jeho rozšíření
Název práce (anglicky):	Variational Autoencoder and its Extensions

Pokyny pro vypracování:

1. Seznamte se s teorií odhadu aposteriorní distribuce neznámých parametrů modelu z dat a jejími aproximacemi. Zvláštní pozornost věnujte metodám založeným na minimalizaci Kullback-Leiblerovy divergence ve formě Variačního Bayese a ELBO.
2. Na jednoduchém modelu lineární regrese s neznámými hyperparametry apriorního rozdělení demonstrujte vlastnosti aproximací ELBO a Variační Bayes.
3. Seznamte se s neuronovou sítí jako s univerzální aproximací spojité funkce. Uvažujte model transformace standardního normálního náhodného rozložení pomocí neuronové sítě. Odvoďte ELBO odhad jeho parametrů za předpokladu, že aposteriorní distribuce je opět normální se střední hodnotou a variancí modelovanou opět neuronovou sítí. Diskutujte souvislost výsledku (VAE) s topologií neuronové sítě známou jako autoencodér.
4. S využitím existujícího kódu demonstrujte vlastnosti VAE na jednoduchých příkladech i na vhodně zvolených reálných datech. Diskutujte výhody a nevýhody metody.
5. Proveďte rešerši literatury metod navazujících na VAE a jejich principů. Diskutujte rozšíření pomocí Wassersteinových divergencí, VAMP aprioren a dalších metod. Výsledky demonstrujte na příkladech z předchozího bodu.
6. Naimplementujte vybrané rozšíření VAE z literatury a demonstrujte jeho vlastnosti na jednoduchých příkladech a na zvolených reálných datech.

Doporučená literatura:

1. Ch. M. Bishop, Pattern recognition and machine learning. Springer, 2006.
2. D. P. Kingma, M. Welling, Auto-encoding variational bayes. ArXiv preprint arXiv:1312.6114, 2013.
3. I. Tolstikhin, 0. Bousquet, S. Gelly, B. Schoelkopf, Wasserstein auto-encoders. ArXiv preprint arXiv:1711.01558, 2017.
4. J. M. Tomczak, M. Welling, VAE with a VampPrior. ArXiv preprint arXiv:1705.07120, 2017.

Jméno a pracoviště vedoucího bakalářské práce:

Doc. Ing. Václav Šmídl, Ph.D.
Ústav teorie a informace a automatizace AV ČR, Pod Vodárenskou věží 4, 182 00, Praha 8, Česká republika

Jméno a pracoviště konzultanta:

Datum zadání bakalářské práce: 31.10.2019

Datum odevzdání bakalářské práce: 7.7.2020

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 23. října 2019

garant oboru

vedoucí katedry

děkan

Poděkování:
Chtěla bych poděkovat svému školiteli docentu Šmídlovi za ochotu a trpělivost při vedení mé bakalářské práce. Jeho cenné rady a přátelský přístup byly skvělou motivací. Dále bych chtěla poděkovat své rodině, kteří mě podporují od začátku studia a nikdy to se mnou nevzdávají. Obrovský dík patří Kátě, která mi je oporou v dobrých i zlých časech.

Čestné prohlášení:
Prohlašuji, že jsem tuto práci vypracovala samostatně a uvedla jsem všechnu použitou literaturu.

V Praze dne 24. července 2020 Michaela Mašková

Název práce:

Variační autoencoder a jeho rozšíření

Autor: Michaela Mašková

Obor: Matematické inženýrství

Zaměření: Aplikované matematicko-stochastické metody

Druh práce: Bakalářská práce

Vedoucí práce: Doc. Ing. Václav Šmídl, Ph.D., Ústav teorie informace a automatizace AV ČR

Abstrakt: Variační autoencoder je modelem, který v sobě kombinuje pravděpodobnostní přístup a sílu aproximací pomocí neuronových sítí. Lze jej využít jako generativní model nebo například v detekci anomálií. Jeho výhody i nevýhody, stejně jako jeho možná rozšíření, jsou předmětem bakalářské práce. Popsaný je dále Wassersteinův autoencoder a model s využitím VampPrior. Je odvozeno nové rozšíření směsi variačních autoencoderů s použitím EM algoritmu. Vlastnosti variačního autoencoderu a jeho rozšíření jsou demonstrovány na ukázkových i reálných datech.

Klíčová slova: bayesovské metody, detekce anomálií, neuronové sítě, variační autoencoder

Title:

Variational autoencoder and its extensions

Author: Michaela Mašková

Abstract: Variational autoencoder combines probabilistic approach with powerful estimations via neural networks. The model can be used as a generative model as well as an anomaly detector. Its pros and cons, as well as its extensions, are the subject of this bachelor's thesis. Extensions like Wasserstein autoencoder and a model using VampPrior are described. A new extension using a mixture of variational autoencoders with the usage of EM algorithm is proposed. Properties of a variational autoencoder and its extensions are demonstrated on both artificial and real-world datasets.

Key words: anomaly detection, bayesian methods, neural networks, variational autoencoder

Obsah

Úvod

Bakalářská práce se soustředí na seznámení se s modelem variačního autoencoderu. Variační autoencoder je model strukturou podobný klasickému autoencoderu – skládá se z kodéru a dekodéru. Cílem kodéru je zakódovat informace v datech do latentních proměnných. Úloha dekodéru spočívá v rekonstrukci informace obsažené v latentních proměnných zpět na vstupní data s co nejmenší rekonstrukční chybou. Kodér i dekodér jsou modelovány pomocí neuronových sítí. V klasickém variačním autoencoderu je rozdělení latentních proměnných standardní gaussovo rozdělení. Variační autoencoder lze použít na modelování pravděpodobnostní distribuce dat nebo jako generativní model.

S využitím principu variačních bayesovských metod je odvozena základní rovnice variačního autoencoderu a model je představen na jednoduchých příkladech. Jsou diskutovány výhody i nevýhody původního modelu a představena možná rozšíření. Prvním je Wassersteinův autoencoder, který přidává možnost libovolného výběru rozdělení latentních proměnných. Následuje model s využitím VampPrior, jenž zavádí multimodální apriorní rozdělení.

V rámci inspirace bayesovskými směsovými modely je vytvořeno poslední rozšíření variačního autoencoderu, konkrétně model směsi variačních autoencoderů. Pravděpodobnost příslušnosti do jednotlivých komponent modelu je spočítána pomocí EM algoritmu. Nový model umožňuje popsat data pocházející ze směsi rozdělení, jež jsou navzájem disjunktní. Jeho možné výhody jsou ukázány na jednoduchých ilustračních datech.

Odvozené a popsané modely jsou následně využity pro experimenty v oblasti detekce anomálií. Zkoumané jsou dva reálné datasety. KDD Cup 1999 obsahuje data z internetového provozu a cílem je detekovat nebezpečné připojení. V datasetu HTRU2 je cílem rozeznat mezi kandidáty pulsary.

Kapitola 1

Bayesovské principy odhadu parametrů a neuronové sítě

1.1 Základní pojmy pravděpodobnosti

V bakalářské práci je ve velké míře využívaná teorie pravděpodobnosti a modelování pravděpodobnostních hustot. Pro konzistenci jsou v následující sekci uvedeny pojmy a věty, které budou později využity pro sestavení modelu variačního autoencoderu a jeho rozšíření.

1.1.1 Podmíněná pravděpodobnost a Bayesova věta

V pravděpodobnosti a statistice existují dva různé přístupy k datům a odhadování parametrů, které tato data vysvětlují. Prvním je frekventistický přístup. Jeho cílem je sesbírat reprezentativní vzorek populace a většinou metodou maximální věrohodnosti získat bodový odhad hledaného parametru. Bayesovský přístup do podobného modelu zavádí neurčitost, říká totiž, že odhadovaný parametr není určen jednou hodnotou, ale existuje jako pravděpodobnostní rozdělení s vlastní střední hodnotou, rozptylem a dalšími charakteristikami. Abychom mohli takový model definovat, potřebujeme zavést podmíněnou pravděpodobnost a vyslovit Bayesovu větu [1].

Podmíněnou pravděpodobnost definujeme předpisem

$$\mathrm{P}(X|Y) = \frac{\mathrm{P}(X,Y)}{\mathrm{P}(Y)}, \tag{1.1}$$

kde Y představuje apriorní informaci (předtím, než byl pozorován jev X).

Budeme-li pracovat s úplným souborem jevů $Y = \{Y_i\}_{i=1}^{n}$, kde $\mathrm{P}(Y) = 1$, pak platí věta o úplnosti (*product rule*):

$$\mathrm{P}(X) = \sum_{i=1}^{n} \mathrm{P}(X|Y_i)\mathrm{P}(Y_i) \quad \text{pro diskrétní náhodné veličiny a} \tag{1.2}$$

$$\mathrm{P}(X) = \int \mathrm{P}(X|Y)\mathrm{P}(Y)\mathrm{d}Y \quad \text{pro spojité náhodné veličiny.} \tag{1.3}$$

Díky větě o úplnosti můžeme formulovat tzv. Bayesovu větu

$$\mathrm{P}(Y_i|X) = \frac{\mathrm{P}(X|Y_i)\mathrm{P}(Y_i)}{\sum_{i=1}^{n} \mathrm{P}(X|Y_i)\mathrm{P}(Y_i)} = \frac{\mathrm{P}(X|Y_i)\mathrm{P}(Y_i)}{\mathrm{P}(X)}. \tag{1.4}$$

$$\mathrm{P}(Y|X) = \frac{\mathrm{P}(X|Y)\mathrm{P}(Y)}{\int \mathrm{P}(X|Y)\mathrm{P}(Y)\mathrm{d}Y} = \frac{\mathrm{P}(X|Y)\mathrm{P}(Y)}{\mathrm{P}(X)}. \tag{1.5}$$

1.1.2 Pravděpodobnostní rozdělení

V následující sekci jsou uvedena základní rozdělení pravděpodobnosti společně se základními charakteristikami.

Gaussovo rozdělení

Jedním z nejpoužívanějších pravděpodobnostních rozdělení je Gaussovo rozdělení, jinak označováno jako normální. Pro hodnoty jedné proměnné x definujeme hustotu pravděpodobnosti

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \tag{1.6}$$

kde μ je střední hodnota a σ^2 rozptyl. Udává se dále $\beta = 1/\sigma^2$ jako přesnost.

Pro vícerozměrné rozdělení pak definujeme

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right], \tag{1.7}$$

kde $\boldsymbol{\mu}$ je vektor středních hodnot a $|\boldsymbol{\Sigma}|$ determinant kovarianční matice.

Gamma rozdělení

Definujeme hustotu pravděpodobnosti gamma rozdělení jako

$$\mathcal{G}(x|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}, \tag{1.8}$$

kde gamma funkce je

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} \mathrm{d}x. \tag{1.9}$$

Gamma rozdělení má střední hodnotu a entropii

$$\mathbb{E}[X] = \frac{\alpha}{\beta}, \tag{1.10}$$

$$\mathrm{H}(\mathcal{G}) = \alpha - \ln\beta + \ln\Gamma(\alpha) + (1-\alpha)\psi(\alpha). \tag{1.11}$$

1.1.3 Značení

V rámci celé práce se bude pracovat s víceméně jednotným značením. Kaligrafická písmena $\mathcal{X}, \mathcal{Z}, \ldots$ značí prostor proměnných, velká tiskací písmena $X, Y, \ldots$ náhodné a vstupní proměnné. Distribuční funkce značíme velkým $P(X)$ a jejich příslušné hustoty pravděpodobnosti malými písmeny $p(X)$. Tučná písmena $\mathbf{t}, \mathbf{w}, \ldots$ značí vektory, velká tučná písmena $\mathbf{X}$ pak matice. Označení $\mathbf{0}$ představuje nulový vektor požadované dimenze a $\mathbf{I}$ pak jednotkovou matici.

1.2 Regresní model

1.2.1 Motivační příklad

Vyjdeme z jednoduchého modelového příkladu prokládání bodů křivkou jako v [2]. Označíme vstupní datové body $\mathbf{x} = (x_1, ..., x_n)^T$, které generují hodnoty $\mathbf{t} = (t_1, t_2, \ldots, t_n)^T$ podle předem dané závislosti $t = f(x)$, která je neznámá a zatížená šumem.

Nejjednodušší a přitom poměrně přesnou metodou řešení problému je tzv. metoda nejmenších čtverců. Pracuje s předpokladem, že cílem je minimalizovat kvadrát vzdálenosti jednotlivých bodů od výsledné křivky (závislosti), která je polynomem řádu M. V případě, že $M = 1$, mluvíme o lineární regresi. Definujeme nejdříve funkci

$$y(x, \mathbf{w}) = \sum_{i=0}^{M} w_i x^i \tag{1.12}$$

jako polynom řádu M s koeficienty $\mathbf{w}$. Cílem je získat hodnoty těchto koeficientů. Dále zavádíme chybu metody jako funkci $\mathbf{w}$, kterou chceme minimalizovat

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} [t_n - y(x_n, \mathbf{w})]^2 . \tag{1.13}$$

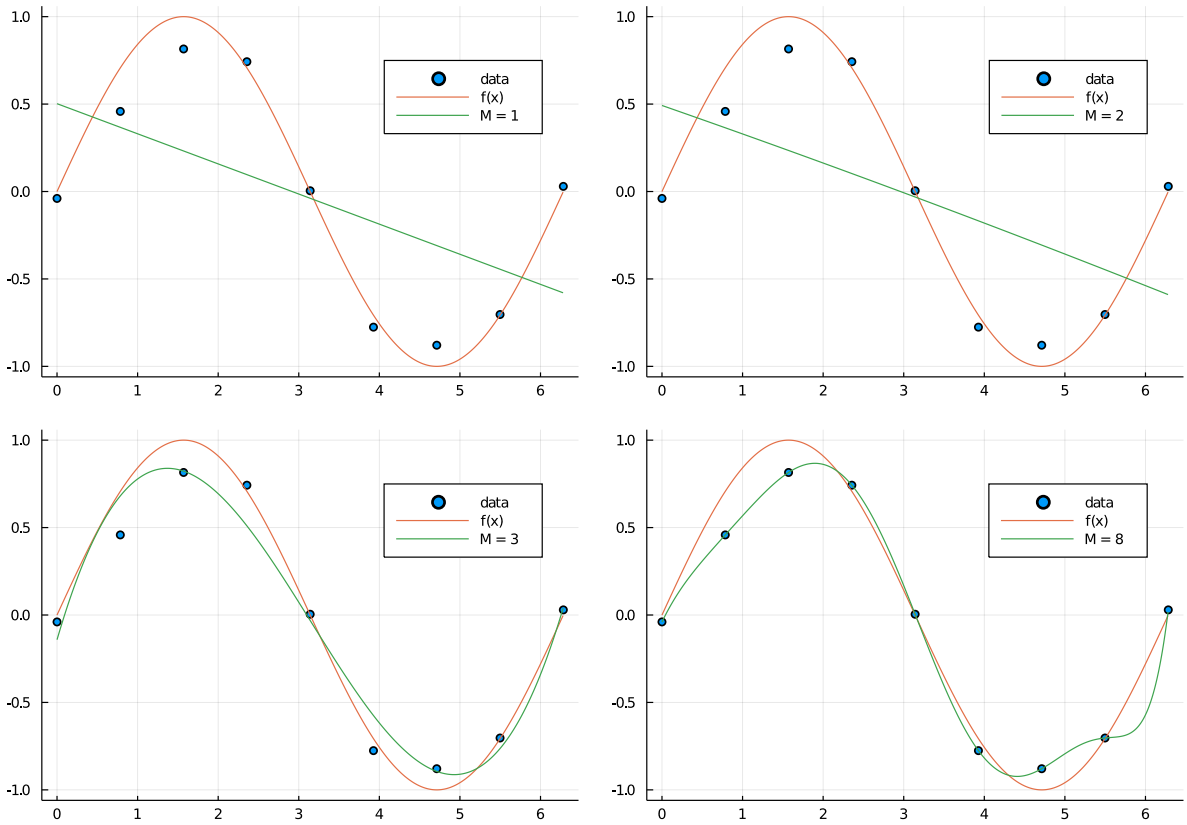
Přesnost metod silně závisí na správném nastavení parametrů modelu, v tomto případě se jedná o řád polynomu M. Pokud bude M malé, model nedokáže zachytit složitost původní závislosti. Naopak pokud bude M příliš velké, může dojít k fenoménu *over-fitting*, kdy je sice přesnost modelu na původních datech velká, nicméně model neodpovídá původní funkci, ze které byly hodnoty generovány.

Pro ilustraci zvažme následující příklad: hodnoty t jsou generovány funkcí $f(x) = \sin(x) + \varepsilon$, kde $\varepsilon \sim \mathcal{N}(0; 0{,}2)$ je náhodný šum. Celkem bylo vygenerováno 9 hodnot a postupně použita metoda nejmenších čtverců pro nastavení parametrů $M \in \{1, 2, 3, 8\}$. Výsledek je na Obr. 1.1. Lze vidět, že pro hodnotu $M = 1$ a $M = 2$ má model příliš málo parametrů a nedokáže zachytit komplexnost původní funkce. Model s $M = 3$ dokáže dobře aproximovat jednotlivé hodnoty a vytvořit závislost podobnou původní funkci. Pokud však zvolíme příliš mnoho parametrů, v tomto případě $M = 8$, dostaneme výslednou křivku, která sice výborně prokládá jednotlivé body, ale nedokáže zachytit obecnou závislost původní funkce.

Podobnou chybu lze odstranit přidáním regularizačního parametru. Obecně totiž platí, že pokud dojde k přefitování, koeficienty w_i budou nabývat hodnot ve vyšších řádech, aby došlo ke kompenzaci rychlého růstu a klesání funkce. Hodnoty lze omezit přidáním regularizačního parametru λ, který má za cíl zmenšit absolutní velikosti koeficientů w_i:

$$E_\lambda(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} [t_n - y(x_n, \mathbf{w})]^2 + \frac{\lambda}{2} ||\mathbf{w}||^2. \tag{1.14}$$

Otázkou je, jak hodnotu λ zvolit. V jednoduchých případech existuje hodnota λ, která se pro daný model jeví jako adekvátní a lze ji nalézt. Existují však případy, kdy nelze najít jednu hodnotu parametru a není jasné, který přístup je nejoptimálnější.



Obrázek 1.1: Fitování metodou nejmenších čtverců pro body generované funkcí $f(x) = \sin(x) + \varepsilon$ vzhledem k nastavení parametru M modelu.

1.2.2 Zobecnění problému

Dále zobecníme případ (1.12) na regresní model, kde je proměnná x transformovaná obecnou nelineární závislostí. K tomu se používají bázové funkce, jednou z nejznámějších je tzv. sigmoid

$$\phi(x) = \frac{1}{1 + \exp(-x)}. \tag{1.15}$$

Pomocí funkcí ϕ_j lze definovat nový regresní model, který zahrnuje nelineární závislost. Definujeme navíc $\phi_0 = 1$, čímž zahrneme samostatný parametr w_0 do kompaktního tvaru modelu. Pak index $j \in \{0, 1, \ldots, M-1\}$, kde M je počet parametrů $\mathbf{w}$. Dostáváme

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}). \tag{1.16}$$

Ztrátová funkce zobecněná na nelineární závislost má pak tvar

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left[t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) \right]^2. \tag{1.17}$$

1.2.3 Bayesovský přístup

Na regresní model dále použijeme bayesovský přístup viz. [2]. Cílem aproximačních metod je predikovat hodnotu t_{N+1} na základě daného x_{N+1} pomocí parametrů $\mathbf{w}$, které byly získány numerickou metodou (například metodou nejmenších čtverců) pomocí původní sady dat

$\mathbf{t} = (t_1, t_2, \ldots, t_N)^T$ a $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$. Nebudeme však chtít hledat bodový odhad hodnoty t, ale pravděpodobnostní rozdělení této proměnné. Předpokládáme-li, že šum ovlivňující konečnou hodnotu t bude mít normální rozdělení, pak i rozdělení t bude normální se střední hodnotou $\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)$ a prozatím neznámou variancí. Rozdělení zapíšeme jako

$$p(t|\mathbf{x}_n, \mathbf{w}, \beta) = \mathcal{N}\left(t|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}\right), \tag{1.18}$$

kde $\beta = \boldsymbol{\Sigma}^{-1}$ je tzv. přesnost a prozatím neznámý parametr.

Budeme-li předpokládat, že jednotlivé realizace t jsou na sobě nezávislé, můžeme sestavit věrohodnostní funkci

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}\left(t_n|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}\right), \tag{1.19}$$

kde N je počet pozorování. Definujeme navíc matici $\boldsymbol{\Phi}$, což je tzv. *design matrix*, pro kterou platí $\boldsymbol{\Phi}_{nj} = \phi_j(\mathbf{x}_n)$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix},$$

kde M je počet parametrů $\mathbf{w}$, tj. matice má rozměr $(N \times M)$.

Věrohodnostní funkci (1.19) lze pak přepsat jako

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \mathcal{N}\left(\mathbf{t}|\boldsymbol{\Phi}\mathbf{w}, \beta^{-1}\mathbf{I}\right). \tag{1.20}$$

Vektor $\mathbf{w}$ lze najít pomocí metody maximální věrohodnosti, tedy maximalizací (1.19). Tím bychom však znovu spočítali pouze bodový odhad koeficientů $\mathbf{w}$. Bayesovský přístup spočívá v hledání pravděpodobnostního rozdělení parametrů $\mathbf{w}$. Rozdělení můžeme navíc rozšířit o tzv. apriorní informaci, která může sloužit například jako regularizační podmínka parametrů. Zde zavedeme podmíněnou apriorní distribuci pro $\mathbf{w}$ přes nový parametr α. Pro jednoduchost zvolíme Gaussovo rozdělení

$$p(\mathbf{w}|\alpha) = \mathcal{N}\left(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}\right). \tag{1.21}$$

Použitím takto definovaného apriorního rozdělení říkáme, že hodnoty $\mathbf{w}$ se mají přibližovat k nule s přesností danou parametrem α. Rozdělení má tak podobnou funkci jako parametr λ používaný v klasické lineární regresi.

Použitím Bayesovy věty (1.5) dostaneme

$$p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \beta, \alpha) \propto p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha). \tag{1.22}$$

Pro zpřehlednění zápisu dále vynecháme závislost na $\mathbf{X}$ a β.

Cílem je zjistit tvar rozdělení $p(\mathbf{w}|\mathbf{t})$, což můžeme provést pomocí porovnání tvarů pravé a levé strany rovnice (1.22). Zapíšeme pouze exponent sdružené hustoty pravděpodobnosti pravé strany (1.22) a výraz upravíme. Je třeba si uvědomit, že hledáme rozdělení v proměnné $\mathbf{w}$, proto členy, které tento vektor neobsahují, jsou zahrnuty do celého rozdělení jako konstanta:

$$\begin{aligned}\ln\left(p(\mathbf{t}|\mathbf{w})p(\mathbf{w}|\alpha)\right) &= -\frac{1}{2}\left(\mathbf{t}-\mathbf{\Phi w}\right)^T\beta\left(\mathbf{t}-\mathbf{\Phi w}\right)-\frac{1}{2}\mathbf{w}^T\alpha\mathbf{I}\mathbf{w}+\text{const.}= \\ &= -\frac{1}{2}\mathbf{t}^T\beta\mathbf{t}+\frac{1}{2}\mathbf{t}^T\beta\mathbf{w}\mathbf{\Phi}+\frac{1}{2}\mathbf{w}^T\mathbf{\Phi}^T\beta\mathbf{t}-\frac{1}{2}\mathbf{w}^T\mathbf{\Phi}^T\beta\mathbf{\Phi w}-\frac{1}{2}\mathbf{w}^T\alpha\mathbf{I}\mathbf{w}+\text{const.} \\ &= -\frac{1}{2}\mathbf{w}^T\mathbf{\Phi}^T\beta\mathbf{\Phi w}+\mathbf{w}^T\mathbf{\Phi}^T\beta\mathbf{t}-\frac{1}{2}\mathbf{w}^T\alpha\mathbf{I}\mathbf{w}+\text{const.}= \\ &= -\frac{1}{2}\mathbf{w}^T\left(\beta\mathbf{\Phi}^T\mathbf{\Phi}+\alpha\mathbf{I}\right)\mathbf{w}+\mathbf{w}^T\mathbf{\Phi}^T\beta\mathbf{t}+\text{const.}\end{aligned} \tag{1.23}$$

Z tvaru exponentu odvození (1.23) je vidět, že se jedná o normální rozdělení, jehož obecný tvar lze zapsat jako

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|m_N, S_N). \tag{1.24}$$

Tvar exponentu (1.24) je

$$-\frac{1}{2}\left(\mathbf{w}-m_N\right)^T S_N^{-1}\left(\mathbf{w}-m_N\right) = -\frac{1}{2}\mathbf{w}^T S_N^{-1}\mathbf{w}+\mathbf{w}^T S_N^{-1}m_N \tag{1.25}$$

a z porovnání (1.23) a (1.25) dostaneme

$$S_N^{-1} = \beta\mathbf{\Phi}^T\mathbf{\Phi}+\alpha\mathbf{I} \tag{1.26}$$

$$m_N = S_N\beta\mathbf{\Phi}^T\mathbf{t}. \tag{1.27}$$

Pokud budeme chtít maximalizovat aposteriorní distribuci (1.24), můžeme maximalizovat její logaritmus a dostáváme tak

$$\ln p(\mathbf{w}|\mathbf{t}) = -\frac{\beta}{2}\sum_{n=1}^{N}\left[t_n-\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right]^2-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}+\text{const.} \tag{1.28}$$

Maximalizace této distribuce pak dává stejný výsledek jako v motivačním příkladu (1.14), když zvolíme parametr jako $\lambda = \alpha/\beta$.

1.3 Neuronové sítě

Neuronovou síť lze popsat jako výpočetní model se skrytými vrstvami, který využívá adaptivního učení parametrů a snaží se o co nejlepší aproximaci trénovacích dat. Používají se v mnoha oblastech, od aproximací funkcí ke složitým klasifikačním úlohám zpracování obrazu.

Neuronová síť má specifickou formu – skládá se z jednotlivých vrstev, které na sebe navazují prostřednictvím tzv. neuronů, uzlů sítě. Myšlenka neuronové sítě kopíruje skutečnou biologickou síť neuronů v našem těle. Jako zakladatelé této myšlenky jsou uváděni Walter Pitts a Warren McCulloch, logik a neurobiolog, kteří společně publikovali článek *A logical calculus of the ideas immanent in nervous activity* v časopise *the Bulletin of Mathematical Biophysics 5:115-133* v roce 1943. V něm zavádí zjednodušený model neuronu.

Architektura neuronové sítě se tak skládá z neuronů a spojení mezi nimi, které jsou realizovány pomocí bázových funkcí (nejčastěji nelineárních). Důležitost jednotlivých spojů je upravována pomocí vah. Právě tyto váhy fungují jako adaptivní parametry neuronové sítě, jejich hodnoty jsou předmětem učení na trénovacích datech.

Matematickou strukturu neuronové sítě definujeme jako v [2]. Základní vrstva neuronové sítě má tvar

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \tag{1.29}$$

kde D je počet proměnných, které vstupují do modelu, $j \in 1, \ldots, M$, kde M je počet parametrů w. Parametru w_{j0} se říká práh (v anglické literatuře *bias*). V biologickém modelu tento práh představuje hodnotu, kterou je nutno překročit, aby došlo k aktivaci daného neuronu.

Hodnoty a_j se nazývají aktivace a každá z nich je dále transformovaná nelineární a diferencovatelnou funkcí $h(.)$ tak, že získáme novou proměnnou

$$z_j = h(a_j). \tag{1.30}$$

Hodnoty z_j jsou skryté proměnné. Jako aktivační funkce h se volí například sigmoid, ReLU funkce nebo funkce swish.

Proměnné z_j lze dále vnořit do další vrstvy tak, abychom na konci získali aktivace

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}, \tag{1.31}$$

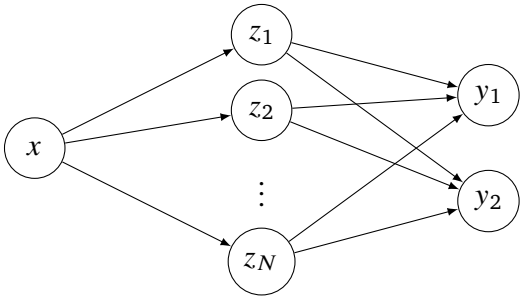
kde $k \in 1, \ldots, K$ a K je celkový počet výstupních proměnných. Horní index přestavuje pořadí dané vrstvy neuronové sítě.

Pokud vnoříme *bias* parametry dovnitř sumy tak, že definujeme $x_0 = 1$, dostaneme kompaktní model, který má následující podobu

$$y_k(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right). \tag{1.32}$$

V takovém modelu vidíme jasně, že má dvě vrstvy. Počet vrstev a počet neuronů v nich je možné kombinovat nejrůznějšími způsoby podle potřeb modelu.

Vizualizaci jednoduchého modelu neuronové sítě můžeme vidět na Obr. 1.2. Do modelu vstupuje jedna proměnná x, která je transformovaná pomocí vah $w_{1:N,1}^{(1)}$ s aktivační funkcí $h_1(.)$ do podoby proměnných $z_1, \ldots, z_N$. Proměnné z_i jsou dále transformovány vahami $w_{1:2,1:N}$ a aktivační funkcí $h_2(.)$ do podoby dvou výstupů y_1 a y_2.



Obrázek 1.2: Základní schéma neuronové sítě s jedním vstupem, N neurony ve skryté vrstvě a dvěma výstupy.

1.3.1 Ztrátová funkce a back-propagation

Důležitou součástí neuronových sítí je jejich ztrátová funkce ozn. $L(\mathbf{w})$ jako funkce parametrů $\mathbf{w}$. Udává, jak moc se model neuronové sítě liší od správných hodnot/rozhodnutí. Právě díky ní a algoritmu zvaném *back-propagation* (neboli zpětná propagace) [2] je možné v každé iteraci upravit hodnoty vah tak, aby se neuronová síť učila co nejlépe aproximovat trénovací data.

Nejjednodušším způsobem jak trénovat neuronovou síť je použít gradientní sestup. Parametry $\mathbf{w}$ upravíme o malý krok ve směru negativního gradientu, tj.

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla L(\mathbf{w}_k). \tag{1.33}$$

Parametr η se nazývá *learning rate* a určuje velikost kroku. Jeho správné nastavení je klíčové pro konvergenci modelu.

Abychom mohli upravit každou jednotlivou váhu w_{ij}, potřebujeme použít zpětnou propagaci. Využijeme proto derivaci složené funkce, abychom se dostali do potřebné hloubky. Parciální derivace ztrátové funkce podle dané váhy bude tak mít tvar

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}. \tag{1.34}$$

Označíme $\frac{\partial L}{\partial a_j} = \delta_j$ a aplikujeme řetězové pravidlo derivací

$$\delta_j = \sum_k \frac{\partial L}{\partial a_k} \frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \frac{\partial}{\partial a_j} \sum_i w_{ki} z_i = \sum_k \delta_k \frac{\partial}{\partial a_j} \sum_i w_{ki} h(a_i). \tag{1.35}$$

Jelikož derivace podle a_j je nenulová jen pro $i = j$, dostáváme tak rovnici zpětné propagace

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k. \tag{1.36}$$

Využijeme značení (1.30). Výraz (1.34) tak dostane tvar

$$\frac{\partial L}{\partial w_{ji}} = \delta_j z_i. \tag{1.37}$$

Učení neuronové sítě pak probíhá ve dvou krocích. V prvním dojde k dopředné propagaci, kdy vypočítáme hodnotu ztrátové funkce v daném bodě x. Následuje zpětná propagace, výpočet jednotlivých gradientů a upravení vah pomocí (1.33).

Stochastic gradient descent

V ideálním případě by byla zpětná propagace provedena pro všechny jednotlivé body zároveň. Takovým způsobem bychom se však dostali na obrovskou výpočetní náročnost. Z tohoto důvodu se při výpočtu gradientů používá tzv. *stochastic gradient descent*, nebo-li metoda stochastického gradientu. Pracuje s myšlenkou, že celková hodnota ztrátové funkce je součtem ztrát pro jednotlivé body, tedy

$$L(\mathbf{w}) = \sum_{n=1}^{N} L_n(\mathbf{w}). \tag{1.38}$$

Název stochastický plyne z faktu, že v každé iteraci je náhodně vybrán datový bod, spočítá se gradient jeho ztrátové funkce a váhy jsou upraveny na základě tohoto gradientu jako

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla L_n(\mathbf{w}_k). \tag{1.39}$$

Jedna z výhod oproti klasickému gradientu je fakt, že je jednodušší dostat se z lokálního minima směrem k minimu globálnímu. Lokální minimum totiž nebude minimem pro všechny body a stochastický gradient umožní posun jiným směrem.

Další optimalizační metody

V rámci vylepšování neuronových sítí došlo i k vylepšení trénovacích algoritmů použitím lepších optimalizačních nástrojů. Patří mezi ně například RMSProp nebo ADAM [3]. Oba nástroje používají adaptivní parametr η místo jednoho pevného čísla, navíc jsou schopny počítat různé η_θ pro konkrétní parametry. V každé dimenzi tak může být η parametr rozdílný.

ADAM využívá pouze gradient prvního řádu a jeho formule pro změnu parametru θ je následující

$$\begin{aligned}
\theta_t &= \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \quad \text{kde} \\
g_t &= \nabla_\theta f_t(\theta_{t-1}), \\
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \quad \text{a} \quad \hat{m}_t = \frac{m_t}{1-\beta_1^t}, \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad \text{a} \quad \hat{v}_t = \frac{v_t}{1-\beta_2^2}.
\end{aligned}$$

Parametry β_1, β_2 a ϵ jsou hyperparametry optimalizátoru. Jejich nastavení je možné měnit, autoři uvádí doporučené hodnoty, které jsou dobrým úvodním nastavením pro většinu experimentů.

1.3.2 Aktivační funkce

Aktivační funkce jsou základním stavebním kamenem neuronových sítí a jsou to ony, které umožňují jejich aplikovatelnost na řešení rozličných problémů. V minulosti se nejčastěji používala jako aktivační funkce již zmíněná funkce sigmoid, případně funkce tanh. Později se objevily další aktivační funkce, díky nimž přesnost neuronových sítí znovu vzrostla.

ReLU je definována jako $f(x) = \max(0, x)$. První zmínka o jejím využití je z roku 2000, nicméně do širšího povědomí se dostala až mezi roky 2009 a 2011 [4], kdy bylo poprvé zkoumáno její využití v hlubokých neuronových sítích. ReLU vytváří pravou řídkou neuronovou síť s nulovými výstupy určitých neuronů, kde při náhodné inicializaci je poté aktivováno pouze zhruba 50 % skrytých proměnných. Nulové výstupy umožňují přetržení daného informačního vlákna, tedy zakódovat fakt, že daný faktor nepřidává informaci následujícímu neuronu. Díky jejímu zavedení bylo dosaženo rychlého vývoje v komplexních neuronových sítích. Výhoda ReLU může však být zároveň i nevýhodou, kdy může vzniknout příliš mnoho nulových neuronů. Navíc v bodě nula neexistuje její gradient.

Nevýhody ReLU vedly k výzkumu nových aktivačních funkcí. V článku [5] autoři nalezli hned několik kandidátů na aktivační funkce, které by mohly zajistit lepší výsledky než ReLU. Nejlepší z nich se ukázala funkce swish definovaná jako $f(x) = x \cdot \sigma(\beta x)$, kde β lze získat z dat. Swish je oproti ReLU diferencovatelná v každém bodě a její gradient je nenulový i pro případy kdy $x < 0$.

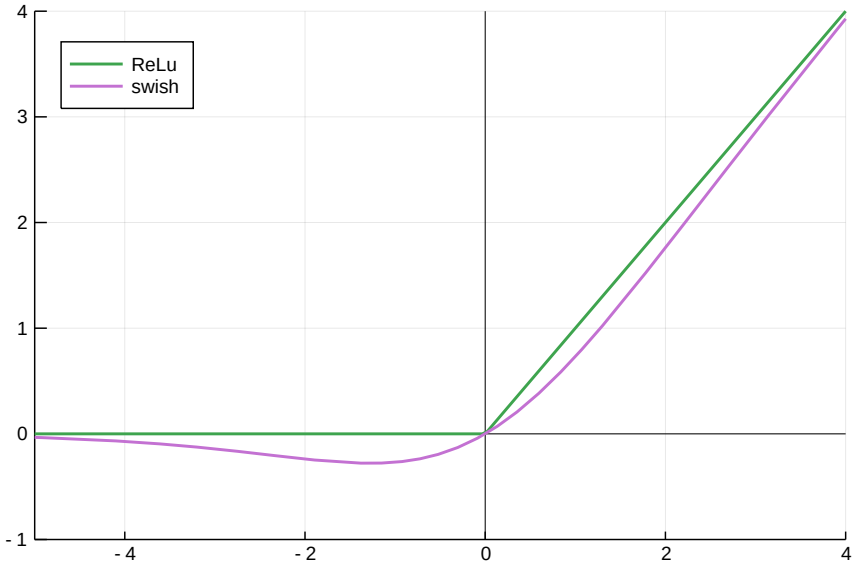
Grafické porovnání zmíněných aktivační funkcí lze nahlédnout na Obr. 1.3.

Obě zmíněné aktivační funkce mají nevýhodu oproti sigmoidu a tanh, jelikož nejsou omezené a pro data vysokých řádů se mohou rychle dostat do nekonečna. Možným řešením je použít regularizaci.

1.3.3 Julia Language a Flux.jl

V rámci bakalářské práce byl pro vytvoření všech modelů použit jazyk Julia s využitím balíčku Flux.jl [6], který byl vytvořen primárně pro modelování a trénování neuronových sítí. Julia jako vysokoúrovňový jazyk umožňuje snadné definování modelů a přibližuje se klasickému matematickému vyjadřování.

V rámci práce s neuronovými sítěmi je zde zmíněno několik základních příkazů pro definování neuronových sítí a modelů.



Obrázek 1.3: Porovnání aktivačních funkcí ReLU a swish v okolí bodu 0.

Příkaz `Dense(in,out,activation)` vytvoří jednu vrstvu neuronové sítě s `in` počtem vstupních neuronů, `out` počtem výstupních neuronů a aktivační funkcí `activation`, kam je možné dosadit například funkci sigmoid, ReLU nebo swish. `Dense` vrstva odpovídá formulaci (1.29), respektive výrazu $y = W \cdot x + b$. Pomocí příkazu `Chain` lze jednotlivé vrstvy řetězit stejně jako v (1.31). Neuronová síť z obrázku 1.2 tak může být v jazyce Julia definovaná jako

```
NN(x) = Chain(Dense(2,N,swish),Dense(N,2,sigmoid)),
```

což vytvoří neuronovou síť se vstupem dimenze 1, jednou vnitřní vrstvou s N neurony, která je transformovaná aktivační funkcí swish, a nakonec výstupem o dimenzi 2, který je také transformován, tentokrát aktivační funkcí sigmoid.

Kapitola 2

Variační autoencoder a jeho rozšíření

2.1 Variační Bayesovské metody

Předmětem bakalářské práce je využít různé modely založené na pravděpodobnostním přístupu k hledání tzv. latentních proměnných. Latentní proměnné jsou proměnné, které získáváme modelováním pozorovaných proměnných. Je možné je nazývat také vysvětlující proměnné, protože jimi lze popsat reálná data.

Snažíme se pak maximalizovat pravděpodobnost, že data X pocházejí z generativního procesu definovaného rovnicí

$$p(X) = \int_{\mathcal{Z}} p(X|Z)p(Z)\mathrm{d}Z. \tag{2.1}$$

Rovnice (2.1) říká, že se snažíme najít takové proměnné Z, které nejlépe vysvětlují data X – poté bude pravděpodobnost $p(X)$ maximalizována. V reálných modelech se ovšem velmi často stává, že integrál (2.1) není analyticky spočitatelný.

2.1.1 ELBO a Kullback-Leiblerova divergence

Evidence lower bound

Nechť X jsou pozorovaná data a naším cílem je najít latentní proměnné Z tato data popisující, tedy rozdělení $p(Z|X)$. Použijeme-li Bayesův teorém na příslušné pravděpodobnosti, dostaneme

$$p(Z|X) = \frac{p(X|Z)p(Z)}{p(X)} = \frac{p(X|Z)p(Z)}{\int_{\mathcal{Z}} p(X,Z)\mathrm{d}Z}. \tag{2.2}$$

Jak už bylo zmíněno, integrál $p(X)$ nemusí být analyticky spočitatelný. Zavedeme proto novou distribuci $q(Z)$, která má hledané $p(Z|X)$ co nejlépe aproximovat. Tato nová hustota může záviset na svých vlastních parametrech θ, které se dále snažíme optimalizovat tak, aby byl rozdíl $q(Z)$ a $p(Z|X)$ co nejmenší. Narozdíl od $p(Z|X)$ by mělo mít $q(Z)$ jednodušší formu.

Pro následující odvození využijeme Jensenovu nerovnost [2] str. 56. Tato věta říká, že pokud je funkce f konkávní, platí

$$f[\mathbb{E}(x)] \geq \mathbb{E}[f(x)]. \tag{2.3}$$

Vyjdeme z distribuce $p(X)$, využijeme Jensenovu nerovnost (2.3), jelikož logaritmus je konkávní funkce, a dostaneme tzv. *evidence lower bound* nebo také *variational lower bound* [7]:

$$\begin{aligned}\ln p(X) &= \ln \int_Z p(X,Z) = \\ &= \ln \int_Z p(X,Z)\frac{q(Z)}{q(Z)} = \\ &= \ln \left(\mathbb{E}_q \left[\frac{p(X,Z)}{q(Z)} \right] \right) \geq \\ &\geq \mathbb{E}_q \left(\ln \frac{p(X,Z)}{q(Z)} \right) = \\ &= \int_Z q(Z) \ln \frac{p(X,Z)}{q(Z)} = \mathcal{L}(q).\end{aligned} \tag{2.4}$$

Evidence lower bound neboli ELBO tedy definujeme jako

$$\mathcal{L}(q) = \int_Z q(Z) \ln \frac{p(X,Z)}{q(Z)}. \tag{2.5}$$

Výsledek implikuje, že pokud je cílem maximalizovat pravděpodobnost $p(X)$, lze problém aproximovat maximalizací ELBO.

Kullback-Leiblerova divergence

Kullback-Leiblerova divergence [8] se používá pro měření rozdílnosti dvou pravděpodobnostních distribucí stejné proměnné x – $p(x), q(x)$. KL divergence je úzce spjatá s teorií informace, kde nachází využití jako míra ztracené informace, pokud je distribuce $p(x)$ aproximována $q(x)$.

Nechť jsou $p(x)$ a $q(x)$ hustoty pravděpodobnosti. Pak Kullback-Leiblerova divergence $\mathcal{D}_{KL}\left(p(x)||q(x)\right)$ je definována jako

$$\mathcal{D}_{KL}\left(p(x)||q(x)\right) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)} \tag{2.6}$$

pro diskrétní případ a

$$\mathcal{D}_{KL}\left(p(x)||q(x)\right) = \int_{-\infty}^{\infty} p(x) \ln \frac{p(x)}{q(x)} \mathrm{d}x \tag{2.7}$$

pro spojitou proměnnou x.

Důležité je zmínit vlastnosti $\mathcal{D}_{KL}$:

1. $\mathcal{D}_{KL}$ není symetrická, jak je vidět přímo z definice, tj. $\mathcal{D}_{KL}(p||q) \neq \mathcal{D}_{KL}(q||p)$,
2. $\mathcal{D}_{KL}$ je nezáporná, tj. $\mathcal{D}_{KL}(p||q) \geq 0$ a $\mathcal{D}_{KL}(p||q) = 0$ právě tehdy, když $p(x) = q(x)$.

Jelikož už jsou potřebné pojmy definované, podíváme se dále na vztah ELBO a KL divergence.

$$\begin{aligned}&\mathcal{D}_{KL}\left[q(Z)||p(Z|X)\right] = \int_Z q(Z) \ln \frac{q(Z)}{p(Z|X)} = -\int_Z q(Z) \ln \frac{p(Z|X)}{q(Z)} = \\ &= -\int_Z q(Z) \ln \frac{p(X,Z)}{q(Z)p(X)} = \int_Z q(Z) \ln p(X) - \int_Z q(Z) \ln \frac{p(X,Z)}{q(Z)} = \\ &= \ln p(X) \int_Z q(Z) - \int_Z q(Z) \ln \frac{p(X,Z)}{q(Z)} = \ln p(X) - \int_Z q(Z) \ln \frac{p(X,Z)}{q(Z)} = \\ &= \ln p(X) - \mathcal{L}(q).\end{aligned} \tag{2.8}$$

Z odvození dostáváme vztah

$$\ln p(X) = \mathcal{D}_{KL}\left[q(Z)||p(Z|X)\right] + \mathcal{L}(q). \tag{2.9}$$

Ve vztahu $\ln p(x) \geq \mathcal{L}(q)$ tak nastává rovnost, pokud je KL divergence mezi $q(Z)$ a $p(Z|X)$ rovna nule.

ELBO lze rozepsat i následujícím způsobem:

$$\begin{aligned}
\mathcal{L}(q) &= \int_Z q(Z) \ln \frac{p(X,Z)}{q(Z)} = \\
&= \int_Z q(Z) \ln \frac{p(X|Z)p(Z)}{q(Z)} = \\
&= \int_Z q(Z) \ln \frac{p(Z)}{q(Z)} + \int_Z q(Z) \ln p(X|Z) = \\
&= \int_Z q(Z) \ln p(X|Z) - \int_Z q(Z) \ln \frac{q(Z)}{p(Z)} = \\
&= \mathbb{E}_q\left[\ln p(X|Z)\right] - \mathcal{D}_{KL}\left[q(Z)||p(Z)\right].
\end{aligned} \tag{2.10}$$

Z kombinace (2.9) a (2.10) dostáváme

$$\ln p(X) - \mathcal{D}_{KL}\left[q(Z)||p(Z|X)\right] = \mathbb{E}_q\left[\ln p(X|Z)\right] - \mathcal{D}_{KL}\left[q(Z)||p(Z)\right]. \tag{2.11}$$

Tato rovnice v lehce upraveném tvaru je pak základem variačního autoencoderu.

2.1.2 Konkrétní příklad

Řekněme, že se budeme chtít vrátit k původnímu příkladu lineární regrese s použitím apriorní informace i o parametru α. Budeme tedy hledat sdruženou hustotu pravděpodobnosti

$$p(\mathbf{w}, \alpha|\mathbf{X}, \mathbf{t}, \beta) = p(\mathbf{w}|\mathbf{X}, \mathbf{t}, \alpha, \beta)p(\alpha|a, b). \tag{2.12}$$

Rozdělení parametrů $\mathbf{w}$ volíme opět normální, rozdělení α pak gamma rozdělení

$$p(\mathbf{w}|\alpha, t, \beta) = \mathcal{N}\left(\mathbf{w}|\mathbf{0}, \alpha\mathbf{I}\right) \tag{2.13}$$

$$p(\alpha|a, b) = \Gamma(\alpha|a_0, b_0). \tag{2.14}$$

Tato rozdělení aproximujeme rozděleními q následovně

$$q(\mathbf{w}|\alpha, t, \beta) = \mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}}, \Sigma_w\right) \tag{2.15}$$

$$q(\alpha|a, b) = \Gamma(\alpha|a, b). \tag{2.16}$$

Výhoda spočívá v tom, že rozdělení q jsou vzájemně nezávislá, tj. $q(\mathbf{w}, \alpha) = q(\mathbf{w})q(\alpha)$.

Použijeme dále KL divergenci mezi sdruženým rozdělením $p(\mathbf{w}, \alpha)$ a $q(\mathbf{w}, \alpha)$, kterou budeme chtít ve výsledku minimalizovat, abychom získali co nejpřesnější aproximaci:

$$\begin{aligned}
&\mathcal{D}_{KL}\left[q(\mathbf{w}, \alpha)||p(\mathbf{w}, \alpha)\right] = \\
&= \mathcal{D}_{KL}\left[\mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}}, \Sigma_w\right)\mathcal{G}(\alpha|a, b)||\mathcal{N}\left(\mathbf{w}|\mathbf{0}, \alpha\mathbf{I}\right)\mathcal{G}(\alpha|a_0, b_0)\right] = \\
&= \mathbb{E}_q\left[\ln \mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}}, \Sigma_w\right) + \ln \mathcal{G}(\alpha|a, b) - \ln \mathcal{N}\left(\mathbf{w}|\mathbf{0}, \alpha\mathbf{I}\right) - \ln \mathcal{G}(\alpha|a_0, b_0)\right].
\end{aligned} \tag{2.17}$$

Jednotlivé výrazy budeme řešit postupně:

$$\begin{aligned}
&\mathbb{E}_q\left[\ln\mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}},\Sigma_w\right)+\ln\mathcal{G}(\alpha|a,b)\right]= \\
&\quad=\mathbb{E}_q\left[\ln\mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}},\Sigma_w\right)\right]+\mathbb{E}_q\left[\ln\mathcal{G}(\alpha|a,b)\right]= \\
&\quad=\iint\mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}},\Sigma_w\right)\mathcal{G}(\alpha|a,b)\ln\mathcal{G}(\alpha|a,b)\mathrm{d}\mathbf{w}\mathrm{d}\alpha \\
&\qquad\quad+\iint\mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}},\Sigma_w\right)\mathcal{G}(\alpha|a,b)\ln\mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}},\Sigma_w\right)\mathrm{d}\mathbf{w}\mathrm{d}\alpha= \\
&\quad=\left[\text{Fubiniho věta, entropie gamma rozdělení (1.11)}\right]= \\
&\quad=-a+\ln b-\ln\Gamma(a)+(a-1)\psi(a)-\frac{1}{2}\ln\left[\det(2\pi\mathrm{e}\Sigma_w)\right]= \\
&\quad=-a+\ln b-\ln\Gamma(a)+(a-1)\psi(a)-\frac{M}{2}\ln 2\pi-\frac{M}{2}-\frac{1}{2}\ln\left[\det\Sigma_w\right]
\end{aligned} \tag{2.18}$$

$$\begin{aligned}
&\mathbb{E}_q\left[\ln\mathcal{N}\left(\mathbf{w}|\mathbf{0},\alpha^{-1}\mathbf{I}\right)\right]= \\
&\quad=\iint\mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}},\Sigma_w\right)\mathcal{G}(\alpha|a,b)\ln\mathcal{N}\left(\mathbf{w}|\mathbf{0},\alpha^{-1}\mathbf{I}\right)\mathrm{d}\mathbf{w}\mathrm{d}\alpha= \\
&\quad=\int\mathrm{d}\alpha\mathcal{G}(\alpha|a,b)\int\mathrm{d}\mathbf{w}\mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}},\Sigma_w\right)\ln\mathcal{N}\left(\mathbf{w}|\mathbf{0},\alpha^{-1}\mathbf{I}\right)= \\
&\quad=\int\mathrm{d}\alpha\mathcal{G}(\alpha|a,b)\mathbb{E}_{\mathcal{N}(\mathbf{w}|\hat{\mathbf{w}},\Sigma_w)}\left[-\frac{M}{2}\ln\frac{2\pi}{\alpha}-\frac{1}{2}\mathbf{w}^T\alpha\mathbf{I}\mathbf{w}\right]= \\
&\quad=\int\mathrm{d}\alpha\mathcal{G}(\alpha|a,b)\left[-\frac{M}{2}\ln\frac{2\pi}{\alpha}-\frac{1}{2}\mathrm{Tr}(\alpha\mathbf{I}\Sigma_w)-\frac{1}{2}\hat{\mathbf{w}}^T\alpha\mathbf{I}\hat{\mathbf{w}}\right]= \\
&\quad=-\frac{M}{2}\ln 2\pi+\frac{M}{2}\int\mathcal{G}(\alpha|a,b)\ln\alpha\mathrm{d}\alpha-\frac{1}{2}\mathrm{Tr}(\alpha\mathbf{I}\Sigma_w)-\frac{1}{2}\int\mathcal{G}(\alpha|a,b)\hat{\mathbf{w}}^T\alpha\mathbf{I}\hat{\mathbf{w}}\mathrm{d}\alpha= \\
&\quad=-\frac{M}{2}\ln 2\pi+\frac{M}{2}\left(\psi(a)-\ln b\right)-\frac{1}{2}\mathrm{Tr}\left(\frac{a}{b}\mathbf{I}\Sigma_w\right)-\frac{1}{2}\hat{\mathbf{w}}^T\frac{a}{b}\mathbf{I}\hat{\mathbf{w}}
\end{aligned} \tag{2.19}$$

$$\begin{aligned}
&\mathbb{E}_q\left[\ln\mathcal{G}(\alpha|a_0,b_0)\right]= \\
&\quad=\iint\mathcal{N}\left(\mathbf{w}|\hat{\mathbf{w}},\Sigma_w\right)\mathcal{G}(\alpha|a,b)\ln\mathcal{G}(\alpha|a_0,b_0)\mathrm{d}\mathbf{w}\mathrm{d}\alpha=\left[\text{Fubiniho věta}\right]= \\
&\quad=\left[\mathcal{G}\left(\alpha|a_0,b_0\right)=\frac{b_0^{a_0}}{\Gamma(a_0)}\alpha^{a_0-1}e^{-b_0\alpha}\right]= \\
&\quad=\int\mathcal{G}(\alpha|a,b)\left[a_0\ln b_0-\ln\Gamma(a_0)+(a_0-1)\ln\alpha-b_0\alpha\right]\mathrm{d}\alpha= \\
&\quad=a_0\ln b_0-\ln\Gamma(a_0)+(a_0-1)\int\mathcal{G}(\alpha|a,b)\ln\alpha\mathrm{d}\alpha-b_0\frac{a}{b}= \\
&\quad=a_0\ln b_0-\ln\Gamma(a_0)+(a_0-1)\left(\psi(a)-\ln b\right)-b_0\frac{a}{b}
\end{aligned} \tag{2.20}$$

Dosazením všech spočítaných výrazů do rovnice (2.17) dostaneme explicitní vyjádření pro výpočet KL divergence mezi $p(\mathbf{w},\alpha)$ a $q(\mathbf{w},\alpha)$.

Získané rovnice lze použít pro odhad parametrů rozdělení pomocí ELBO aproximace. Vlastnosti odhadů demonstrujeme na triviálním příkladu. Nechť modelujeme proměnnou d pomocí parametru m jednoduchou rovnicí

$$d = m + \varepsilon, \tag{2.21}$$

kde ε je náhodný šum, navíc budeme předpokládat, že $\varepsilon \sim \mathcal{N}(0, 1)$. Pak lze zavést rozdělení dat, jehož parametry jsou m a α

$$\begin{aligned} p(d|m,\alpha) &= \mathcal{N}(d|m,1) \\ p(m|\alpha) &= \mathcal{N}(m|0,\alpha^{-1}) \\ p(\alpha) &= \mathcal{G}(\alpha|\gamma_0,\delta_0) \\ p(m,\alpha) &= \mathcal{N}(m|0,\alpha^{-1})\mathcal{G}(\alpha|\gamma_0,\delta_0), \end{aligned}$$

m parametrizuje rozdělení dat, α kontroluje apriorní informaci o m.

Použití ELBO odhadů demonstrujeme zavedením aproximačních pravděpodobností

$$\begin{aligned} q(m) &= \mathcal{N}(m|\hat{m},C) \\ q(\alpha) &= \mathcal{G}(\alpha|\gamma,\delta) \\ q(m,\alpha) &= q(m)q(\alpha) = \mathcal{N}(m|\hat{m},C)\mathcal{G}(\alpha|\gamma,\delta). \end{aligned}$$

Pro ELBO bude platit

$$\mathcal{L}(q(m,\alpha)) = \mathbb{E}_{q(m,\alpha)}\left[\ln p(d|m,\alpha)\right] - \mathcal{D}_{KL}\left[q(m,\alpha)||p(m,\alpha)\right]. \tag{2.22}$$

Výpočet prvního členu vypadá následovně:

$$\begin{aligned} \mathbb{E}_{q(m,\alpha)}\left[\ln p(d|m,\alpha)\right] &= \int \mathcal{N}(m|\hat{m},C)\mathcal{G}(\alpha|\gamma,\delta)\ln\mathcal{N}(d|m,1)\mathrm{d}m\mathrm{d}\alpha = \\ &= \int \mathcal{N}(m|\hat{m},C)\left[-\frac{1}{2}\ln 2\pi - \frac{1}{2}(d-m)^2\right]\mathrm{d}m = \\ &= -\frac{1}{2}\ln 2\pi - \frac{1}{2}d^2 + d\int m\mathcal{N}(m|\hat{m},C)\mathrm{d}m - \frac{1}{2}\int m^2\mathcal{N}(m|\hat{m},C)\mathrm{d}m = \\ &= -\frac{1}{2}\ln 2\pi - \frac{1}{2}d^2 + d\hat{m} - \frac{1}{2}(\hat{m}^2 + c). \end{aligned} \tag{2.23}$$

Druhý člen lze získat dosazením do (2.17). Budeme dále předpokládat, že nemáme žádnou apriorní informaci o parametru α, tudíž $\gamma_0 = \delta_0 = 0$.

Vzniklou funkci získanou dosazením do (2.22) lze optimalizovat pomocí gradientních metod, jedná se o její maximalizaci v parametrech $\hat{m}, S, \gamma$ a δ. Získáváme tak rozdělení $q(m,\alpha)$, kterým aproximujeme $p(m,\alpha)$. Je však nutné podotknout, že maximalizaci ELBO provádíme pro konkrétní hodnotu d. Získáváme tak výsledné distribuce parametrů podmíněné datovým bodem d, $q(m|d)$ a $q(\alpha|d)$.

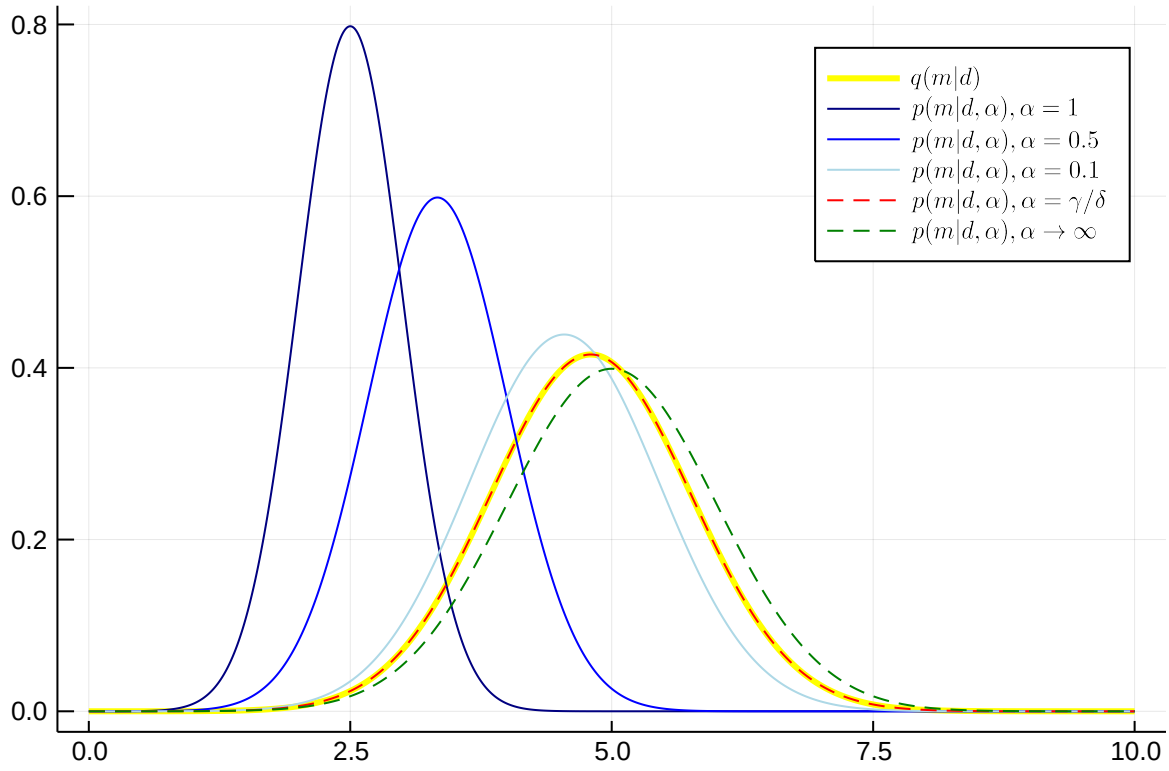
Výhoda aproximační distribuce spočívá v tom, že pro $q(m,\alpha)$ je jednodušší spočítat konečné rozdělení dat d

$$\begin{aligned} p(d) = \int p(d|m,\alpha)p(m,\alpha)\mathrm{d}m\mathrm{d}\alpha \approx \int p(d|m,\alpha)q(m,\alpha)\mathrm{d}m\mathrm{d}\alpha \\ \implies p(d) = \mathcal{N}\left(d|\hat{m},C+1\right) \end{aligned} \tag{2.24}$$

Optimalizace byla provedena pro hodnotu $d = 5$. Výsledné hodnoty parametrů jsou

$$\hat{m} = 4.8$$
$$C = 0.96$$
$$\gamma = 0.5$$
$$\delta = 12.0$$

Cílem příkladu je najít hodnotu m, respektive její pravděpodobnostní rozdělení. Na Obr. 2.1 je vidět porovnání výsledných rozdělení po optimalizaci pomocí ELBO. Distribuce $p(m)$ je zde znázorněna pro různé hodnoty α. Z grafu je možné vidět, že vyšší hodnota parametru α má za následek přibližování m směrem k nule a zpřesňování odhadované hodnoty. Naopak pokud zvolíme $\alpha \to +\infty$, vliv apriorní informace se zcela vytratí.



Obrázek 2.1: Porovnání rozdělení $p(m)$ a $q(m)$ pro případ jedné pozorované hodnoty $d = 5$ a různé parametry α vstupující do $p(m)$.

2.2 Variační autoencoder

Získané poznatky využijeme pro konstrukci modelu variačního autoencoderu [9]. Jeho cílem je naučit se pravděpodobnostní rozdělení dat, které nám umožní vytvářet nové vzorky dat podobné těm původním. Variační autoencoder lze následně využít jako generativní model nebo při modelování dat pomocí latentních proměnných. Abychom mohli variační autoencoder zkonstruovat, vrátíme se zpět k již odvozené rovnici (2.11)

$$\ln p(X) - \mathcal{D}_{KL}\left[q(Z)||p(Z|X)\right] = \mathbb{E}_q\left[\ln p(X|Z)\right] - \mathcal{D}_{KL}\left[q(Z)||p(Z)\right].$$

Další odvození bude sledovat podobnou strukturu jako [10].

Na začátku jsme řekli, že naším cílem je získat rozdělení latentních proměnných v závislosti na datech, tj. $p(Z|X)$. Analytické řešení nemusí existovat, proto jsme zavedli aproximaci pomocí rozdělení $q(Z)$. Aby se tato aproximace co nejlépe blížila původnímu rozdělení, dává smysl, aby i q záviselo na datech X. Pokud navíc zvolíme dostatečně dobrou aproximaci, měl by se výraz $\mathcal{D}_{KL}\left[q(Z|X)||p(Z|X)\right]$ blížit nule a tedy $q(Z|X)$ se bude blížit $p(Z|X)$. Takovou úpravou se rovnice (2.11) změní na

$$\ln p(X) - \mathcal{D}_{KL}\left[q(Z|X)||p(Z|X)\right] = \mathbb{E}_q\left[\ln p(X|Z)\right] - \mathcal{D}_{KL}\left[q(Z|X)||p(Z)\right]. \tag{2.25}$$

Rovnice (2.25) je základem tzv. variačního autoencoderu.

Na levé straně (2.25) dostáváme ELBO, které se stává ztrátovou funkcí. Na pravé straně dostáváme výraz, jenž se snažíme maximalizovat vzhledem k parametrům modelu pomocí gradientních metod.

Otázkou je, jak volit distribuce $q(Z|X)$ a $p(Z)$. Obvyklá volba pro obě hustoty je Gaussovo rozdělení ve tvaru

$$q(Z|X) = \mathcal{N}(Z|\mu(X;\theta), \Sigma(X;\theta)) \tag{2.26}$$

$$p(Z) = \mathcal{N}(Z|\mathbf{0}, \mathbf{I}), \tag{2.27}$$

v čemž spočívá obrovská síla variačního autoencoderu. Rozdělení (2.26) funguje jako kodér a funkce $\mu(X;\theta)$ a $\Sigma(X;\theta)$ jsou modelovány pomocí neuronových sítí, které umožňují, aby se model naučil velmi komplexní strukturu dat. Naopak volba rozdělení $p(Z)$ jako (2.27) nám umožňuje zakódovat informaci do triviálního rozdělení, ze kterého můžeme poté jednoduše vybírat vzorky a modelovat nová data. Rozdělení $p(X|Z)$ funguje jako dekodér a jeho volba může být libovolná. V bakalářské práci je použito normální rozdělení ve tvaru

$$p(X|Z) = \mathcal{N}(X|f(Z,\phi), \omega^2\mathbf{I}), \tag{2.28}$$

kde funkce f je modelovaná neuronovou sítí s vlastními parametry ϕ a ω je hyperparametrem modelu.

Zvolená rozdělení $q(Z|X)$ a $p(Z)$ mají svoji výhodu i v analytické řešitelnosti KL divergence mezi nimi. Pro dvě normální rozdělení $\mathcal{N}(\mu_1, \Sigma_1)$ a $\mathcal{N}(\mu_2, \Sigma_2)$ s dimenzí n lze odvodit

$$
\begin{aligned}
&\mathcal{D}_{KL}\left[\mathcal{N}(\mu_1,\Sigma_1)||\mathcal{N}(\mu_2,\Sigma_2)\right] = \\
&= \mathbb{E}_{\mathcal{N}(\mu_1,\Sigma_1)}\left[\ln\mathcal{N}(\mu_1,\Sigma_1) - \ln\mathcal{N}(\mu_2,\Sigma_2)\right] = \\
&= \mathbb{E}_{\mathcal{N}(\mu_1,\Sigma_1)}\Bigg[-\frac{n}{2}\ln 2\pi - \frac{1}{2}\ln\det\Sigma_1 - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) \\
&\qquad\qquad + \frac{n}{2}\ln 2\pi + \frac{1}{2}\ln\det\Sigma_2 + \frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\Bigg] = \\
&= \mathbb{E}_{\mathcal{N}(\mu_1,\Sigma_1)}\left[\frac{1}{2}\ln\frac{\det\Sigma_2}{\det\Sigma_1} - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right] = \\
&= \frac{1}{2}\ln\frac{\det\Sigma_2}{\det\Sigma_1} - \mathbb{E}_{\mathcal{N}(\mu_1,\Sigma_1)}\left[\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)\right] \\
&\qquad\qquad + \mathbb{E}_{\mathcal{N}(\mu_1,\Sigma_1)}\left[\frac{1}{2}(x-\mu_2)^T\Sigma_2^{-1}(x-\mu_2)\right] = {}^{[11]} \\
&= \frac{1}{2}\ln\frac{\det\Sigma_2}{\det\Sigma_1} - \frac{1}{2}\mathrm{Tr}\left(\Sigma_1^{-1}\Sigma_1\right) - \frac{1}{2}(\mu_1-\mu_1)^T\Sigma_1^{-1}(\mu_1-\mu_1) \\
&\qquad\qquad + \frac{1}{2}\mathrm{Tr}\left(\Sigma_2^{-1}\Sigma_1\right) + \frac{1}{2}(\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2) = \\
&= \frac{1}{2}\ln\frac{\det\Sigma_2}{\det\Sigma_1} - \frac{n}{2} + \frac{1}{2}\mathrm{Tr}\left(\Sigma_2^{-1}\Sigma_1\right) + \frac{1}{2}(\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2) = \\
&= \frac{1}{2}\left[\ln\frac{\det\Sigma_2}{\det\Sigma_1} - n + \mathrm{Tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\mu_1-\mu_2)^T\Sigma_2^{-1}(\mu_1-\mu_2)\right].
\end{aligned}
\tag{2.29}
$$

Aplikováno na konkrétní příklad s (2.26) a (2.27) dostáváme

$$
\mathcal{D}_{KL}\left[\mathcal{N}(Z|\mu(X),\Sigma(X))||\mathcal{N}(Z|\mathbf{0},\mathbf{I})\right] = \frac{1}{2}\left[\mathrm{Tr}\left[\Sigma(X)\right] + \mu(X)^T\mu(X) - \ln\det\Sigma(X) - n\right] \tag{2.30}
$$

Ve variačním autoencoderu se kovarianční matice $\Sigma(X)$ bere jako striktně diagonální, což umožní zjednodušení výpočtu. Výsledná KL divergence se tak díky $(\Sigma)_{ii} = \sigma_i$ upraví na

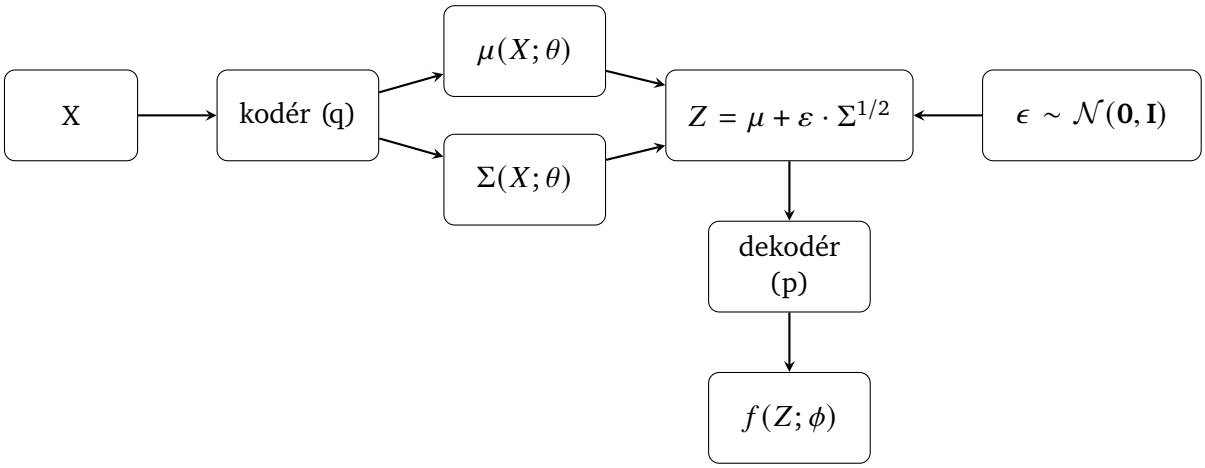
$$
\begin{aligned}
\mathcal{D}_{KL}\left[\mathcal{N}(Z|\mu(X),\Sigma(X))||\mathcal{N}(Z|\mathbf{0},\mathbf{I})\right] &= \frac{1}{2}\left[\sum_i \sigma_i^2 + \sum \mu_i^2 - \ln\prod_i \sigma_i^2 - n\right] = \\
&= \frac{1}{2}\left[\sum_i \sigma_i^2 + \sum \mu_i^2 - \sum_i 2\ln\sigma_i - n\right] = \\
&= \frac{1}{2}\sum_i\left[\sigma_i^2 + \mu_i^2 - 2\ln\sigma_i - 1\right].
\end{aligned}
\tag{2.31}
$$

Zbývá vyřešit první výraz na pravé straně (2.25): $\mathbb{E}_q\left[\ln p(X|Z)\right]$. Jeho problém spočívá v tom, že závisí nejen na parametrech rozdělení p, ale také q. Ve výrazu bychom měli náhodně zvolit Z z rozdělení $\mathcal{N}(\mu(X),\Sigma(X))$. Operace náhodného výběru však nemá gradient, proto není možné na tento výraz použít zpětnou propagaci gradientu.

Problém řeší tzv. reparametrizační trik [9]. Místo abychom náhodně vybírali z rozdělení $\mathcal{N}(\mu(X),\Sigma(X))$, vybereme náhodně proměnnou $\varepsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$, kterou transformujeme pomocí $Z = \mu(X) + \varepsilon\cdot\Sigma^{1/2}(X)$. Ve výsledku tak hledáme gradient ELBO ve tvaru

$$
\mathcal{L} = \mathbb{E}_{\mathcal{N}(0,I)}\left[\ln p\left(X|Z = \mu(X) + \varepsilon\cdot\Sigma^{1/2}(X)\right)\right] - \mathcal{D}_{KL}\left[q(Z|X)||p(Z)\right]. \tag{2.32}
$$

Vizualizace celého procesu je na Obr. 2.2.



Obrázek 2.2: Schéma variačního autoencoderu.

Variační autoencoder se snaží simultánně splnit dvě podmínky určené členy rovnice (2.32). Data vstupují do kodéru modelovaného neuronovými sítěmi μ a Σ. Ty se snaží transformovat vstup a namapovat jej na rozdělení $\mathcal{N}(\mathbf{0}, \mathbf{I})$, k čemuž je použita jako ztrátová funkce KL divergence mezi $q(Z|X)$ a $p(Z)$. Mimo samotný model je vygenerována proměnná $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, která je transformována do proměnné Z reparametrizačním trikem. Proměnná Z následně vstupuje do dekodéru realizovaného funkcí f. Výstup je poté porovnán se vstupem X a funkce f je upravena tak, aby byla minimalizována rekonstrukční chyba, což je v našem případě věrohodnostní funkce ve tvaru

$$\ln p(X|Z) = -\frac{1}{2\omega^2}\left(X - \mu(X) - \varepsilon \cdot \Sigma^{1/2}(X)\right)^2. \tag{2.33}$$

Parametr ω kontroluje vztah mezi oběma podmínkami a určuje, která z nich bude mít větší váhu.

Použití VAE na ilustrační sady dat

Snahou variačního autoencoderu je naučit se distribuci dat. Následně je možné VAE použít jako generativní model. Jednoduše odstraníme kodér a použijeme pouze dekodér, tedy funkci $f(.)$ na náhodně vybrané proměnné z rozdělení $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Pro ilustraci byly vygenerovány vzorky z různě transformovaných normálních rozdělení v jedné a dvou dimenzích. Pro všechny pokusy byla použita podobná architektura neuronové sítě s rozdílnostmi v nastavení počtu neuronů a hyperparametru ω.

Jednotlivé neuronové sítě byly definovány v jazyce Julia následovně:

```
A(x) = Dense(nx,nh,swish)
mu(A), logs(A) = Dense(nh,nz), Dense(nh,nz)
f(z) = Chain(Dense(nz,nh,swish),Dense(nh,nx)).
```

Bod x tedy vstupuje do sítě `A`, která je společným vstupem pro sítě `mu` a `logs`. Dekódovací funkce `f` je pak neuronová síť se dvěma vrstvami. Pro sítě `A` a `f` je použita aktivační funkce swish, `mu` a `logs` mají pouze lineární výstup.

Následují čtyři sady dat se zvyšující se složitostí rozdělení.

1. Nejdříve bylo vygenerováno náhodně 1000 hodnot z rozdělení $\mathcal{N}(5, 3)$. Počet neuronů v síti byl `nx = nh = nz = 1`. Pro každou epochu byl vektor hodnot iterován 1000x. Po každém cyklu trénování bylo z modelu vygenerováno opět 1000 hodnot. Porovnání pravého rozdělení s rozdělením generovaných dat po daném počtu epoch je na Obr. 2.3.

2. Dále se přesuneme do dimenze 2. Počet neuronů je zvýšen na `nx = nh = nz = 2`. Původní data pochází z dvourozměrného normálního rozdělení se střední hodnotou $(3; 5)^T$ a kovariační maticí rovnou jednotkové matici. Na Obr. 2.4 jsou vidět nově vygenerovaná data z natrénovaného modelu po 20 epochách.

3. V dalším případě bylo normální rozdělení z bodu 2. transformováno předpisem:

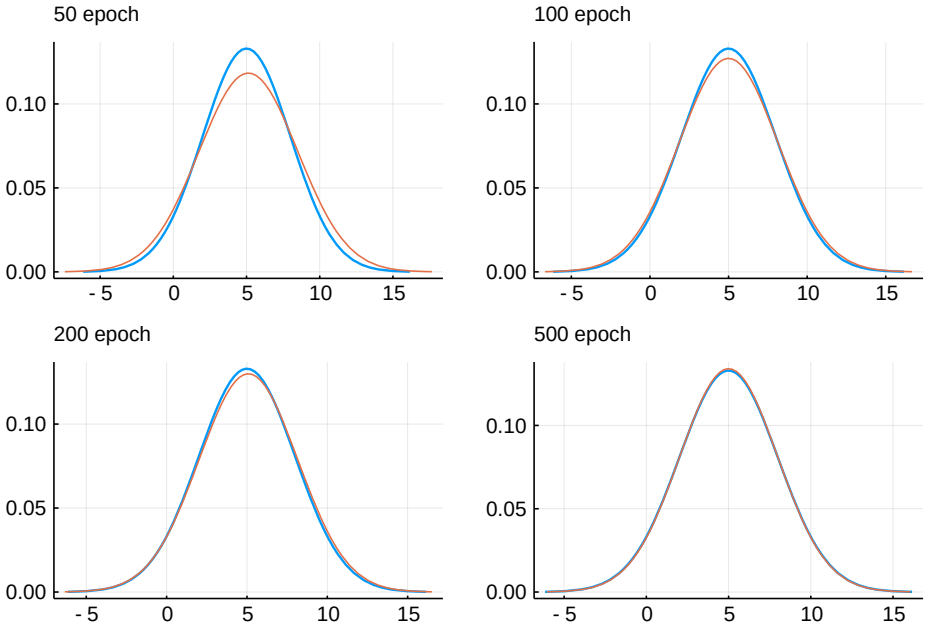
$$x_i' = 2x_i + y_i$$
$$y_i' = x_i + y_i.$$

   Výsledek po 20 epochách je na Obr. 2.5.

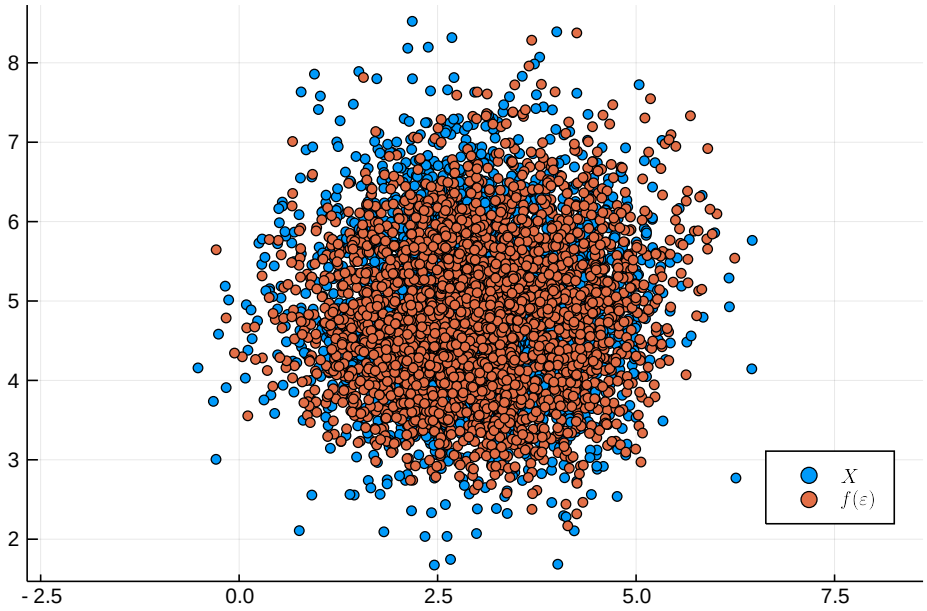
4. Nakonec bylo standardní normální rozdělení ve 2D transformováno předpisem

$$x_i' = \frac{1}{2}x_i + y_i^2$$
$$y_i' = 2y_i$$

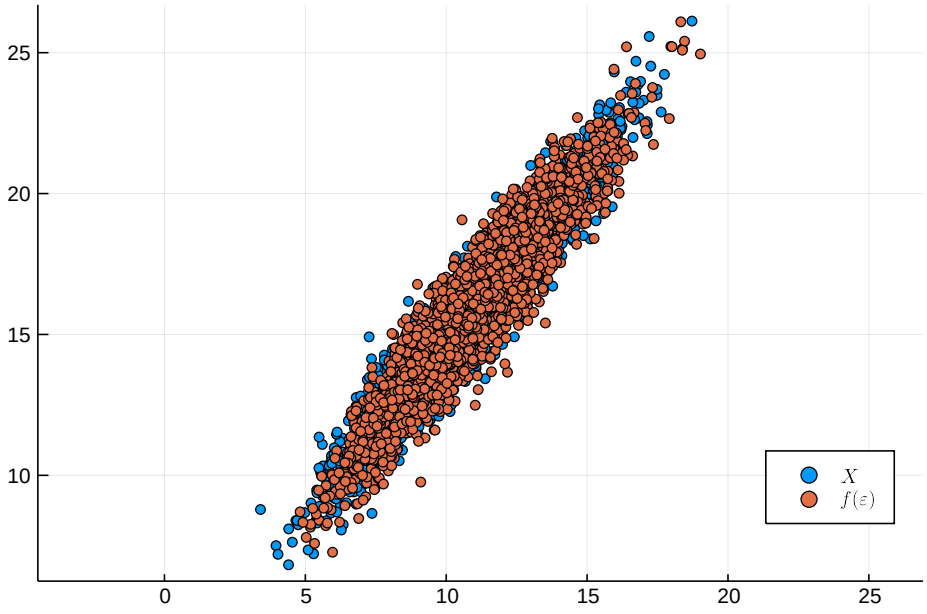
   pro získání složitějšího tvaru. Výsledkem už není normální rozdělení, tedy nemůžeme jeho parametry získat klasickými metodami. Pro natrénování modelu bylo použito nastavení počtu neuronů `nx = 2 = nz, nh = 4`. Menší počet neuronů ve skryté vrstvě se není schopen naučit dostatečně komplexní strukturu dat. Výsledek je vidět na Obr. 2.6.



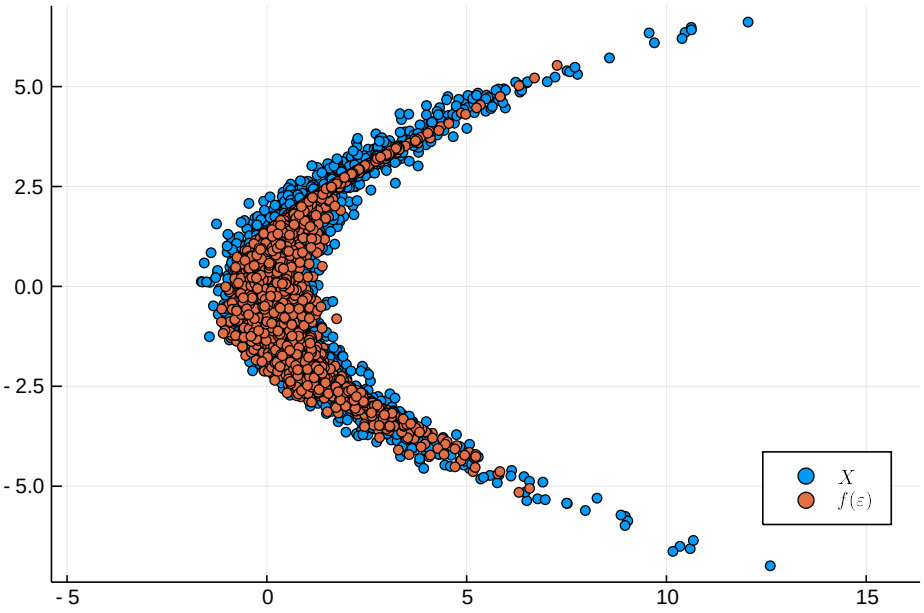
Obrázek 2.3: Porovnání pravé distribuce dat (modrá) s distribucí dat vygenerovaných z modelu VAE po daném počtu epoch.



Obrázek 2.4: Data vygenerovaná natrénovaným modelem VAE po 30 epochách pro data z bodu 2.



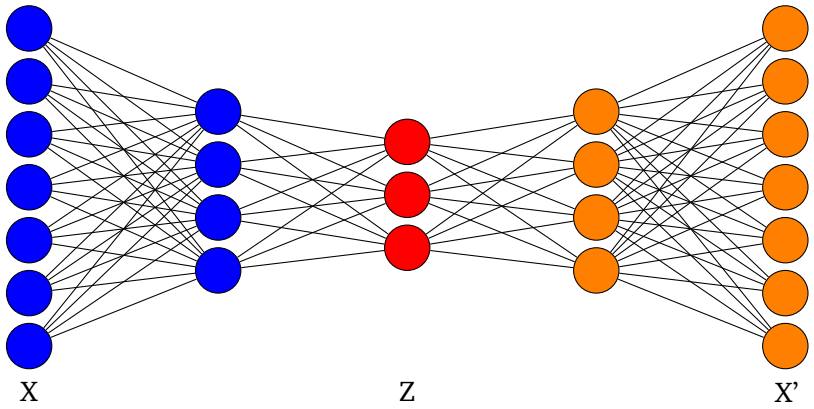
Obrázek 2.5: Data vygenerovaná natrénovaným modelem VAE po 20 epochách pro data z bodu 3.



Obrázek 2.6: Data vygenerovaná natrénovaným modelem VAE po 100 epochách pro data z bodu 4.

2.2.1 Porovnání s klasickými autoencodery

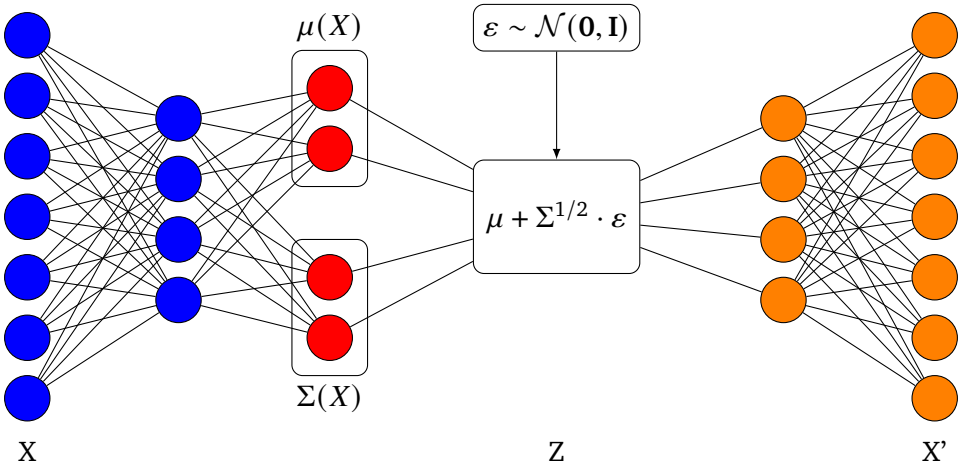
Klasický autoencoder se řadí mezi modely, které se snaží najít reprezentaci dat pomocí neuronových sítí. Jejich struktura je vcelku jednoduchá, grafické znázornění je na Obr. 2.7.



Obrázek 2.7: Grafické znázornění struktury klasického autoencoderu.

Hlavním cílem je komprese informace – data X mohou být reprezentována vektory s vysokou dimenzí, autoencoder tuto informaci zakóduje do latentních proměnných Z, jejichž dimenze je nižší. Optimalizace probíhá formou minimalizace rekonstrukční chyby mezi vstupem a výstupem.

Pro porovnání je na Obr. 2.8 znázorněno grafické schéma variačního autoencoderu. Variační autoencoder je podstatně složitějším modelem, který využívá pravděpodobnostního přístupu, zavádí regularizační člen v podobě KL divergence a místo klasické rekonstrukční chyby používá hodnotu věrohodnostní funkce. Navíc zde probíhá náhodný výběr z $\mathcal{N}(\mathbf{0}, \mathbf{I})$ a reparametrizační trik.



Obrázek 2.8: Grafické znázornění struktury variačního autoencoderu.

2.3 Wassersteinův autoencoder

Wassersteinův autoencoder [12] je dalším druhem autoencoderu, který v sobě spojuje myšlenky variačního autoencoderu a generativních modelů.

Optimalizačním problémem je v tomto případě minimalizace optimální transportní ztráty W_c mezi distribucemi P_X a P_G. P_X představuje reálnou a zároveň neznámou distribuci dat a P_G popisuje latentní model. P_G je určeno distribucí latentních proměnných P_Z a generativním modelem $P_G(X|Z)$. Opět zde počítáme se stejným postupem jako u VAE. V prvním kroku je náhodně vybrán vektor z prostoru $\mathcal{Z}$ pomocí distribuce P_Z, který následně mapujeme na vektor $X \in \mathcal{X}$. Pro příslušné hustoty tak platí

$$p_G(x) = \int_{\mathcal{Z}} p_G(x|z)p_Z(z)\mathrm{d}z, \qquad \forall x \in \mathcal{X}. \tag{2.34}$$

Formulujeme nejdříve optimální transportní problém podle Kantoroviche

$$W_c(P_X, P_G) = \inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y)\sim\Gamma}\left[c(X,Y)\right], \tag{2.35}$$

kde $c(x, y)$ je libovolná nezáporná a měřitelná ztrátová funkce a $\mathcal{P}$ je systém všech sdružených distribucí, které dávají marginální rozdělení P_X a P_G. Podobným trikem jako ve variačním autoencoderu zavádíme distribuci $Q(Z|X)$ tak, že $Q(Z) = \mathbb{E}_{X\sim P_X}\left[Q(Z|X)\right]$ se blíží P_Z. Zbývá už jen doplnit vztah Q a P_G. Věta vyslovená a dokázaná v [12] str. 4 vyslovuje rovnost

$$\inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y)\sim\Gamma}\left[c(X,Y)\right] = \inf_{Q: Q_Z = P_Z} \mathbb{E}_{P_X}\mathbb{E}_{Q(Z|X)}\left[c(X, G(Z))\right], \tag{2.36}$$

kde $G : \mathcal{Z} \to \mathcal{X}$ je libovolná funkce. Není tedy třeba optimalizovat přes všechna X a Y, ale pouze přes náhodné encodery $Q(Z|X)$.

Aby bylo možné hledat Q numericky, zmírníme podmínky kladené na Q a přidáme stejně jako ve VAE penalizační člen v podobě divergence mezi P a Q. Dostáváme tak rovnici Wassersteinova autoencoderu

$$D_{WAE}(P_X, P_G) = \inf_{Q: Q_Z = P_Z} \mathbb{E}_{P_X}\mathbb{E}_{Q(Z|X)}\left[c\left(X, G(Z)\right)\right] + \lambda \cdot \mathcal{D}_Z\left(Q(Z), P(Z)\right). \tag{2.37}$$

Divergenci $\mathcal{D}_Z$ lze volit libovolně a $\lambda > 0$ je hyperparametr.

Kodér Q stejně jako dekodér G modelujeme pomocí neuronových sítí. Jedním z rozdílů mezi VAE a wassersteinovým autoencoderem je možnost u WAE mapovat $\mathcal{X}$ do $\mathcal{Z}$ deterministicky. Volba $p(Z)$ tak může být libovolná, jako příklad můžeme uvést multimodální rozdělení (například směs normálních rozdělení), což klasický variační autoencoder nedokáže. V článku [12] autoři uvádí dvě zkoumané divergence:

1. Jensen-Shannovova divergence $\mathcal{D}_{JS}$, která vede na model založený na generativním postupu,
2. MMD divergence (*maximum mean discrepancy*) [13], která je pro pozitivně definitní kernel $k : \mathcal{Z} \times \mathcal{Z} \to \mathcal{R}$ definovaná jako

$$\mathrm{MMD}_k(P_Z, Q_Z) = \left\| \int_{\mathcal{Z}} k(z, .)\mathrm{d}P_Z(z) - \int_{\mathcal{Z}} k(z, .)\mathrm{d}Q_Z(z) \right\|_{\mathcal{H}_k}. \tag{2.38}$$

Model Wassersteinova autoencoderu je rovnicí (2.37) určen velmi obecně. V této bakalářské práci se budeme zabývat jeho zjednodušeným tvarem

$$\mathcal{L}_W = \mathbb{E}_{\mathcal{N}(0,I)} \left[\ln p \left(X | Z = \mu(X) + \varepsilon \cdot \Sigma^{1/2}(X) \right) \right] + \lambda \cdot \mathrm{MMD}\left[p(Z), q(Z|X) \right], \tag{2.39}$$

kdy bude rekonstrukční chyba ve formě věrohodnostní funkce stejná jako u klasického VAE, ale místo KL divergence aplikujeme MMD vzdálenost.

Použití WAE na ilustrační sady dat

Pro ilustraci byl Wassersteinův autoencoder použit na stejné sady dat jako variační autoencoder. Pro následující jednoduché příklady bylo zvoleno rozdělení $p(Z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, stejně jako u VAE. MMD vzdálenost byla implementována pomocí balíčku IPMeasures.jl v jazyce Julia. Jelikož všechna trénovací data pocházela z normálního rozdělení (případně transformovaného normálního rozdělení), byl zvolen gaussovský kernel s šířkou binu $\gamma = 0,01$, kromě poslední sady dat, kdy byl použit IMQ kernel.

Je nutné podotknout, že pro Wassersteinův autoencoder je nutné volit více hyperparametrů než pro VAE. Jedná se o parametr λ, volbu samotného kernelu k a jeho parametrů (např. šířka binu γ), a následně velikost porovnávaných vzorků. Pro použití MMD vzdálenost je třeba vygenerovat n hodnot z rozdělení $p(Z)$ a tyto porovnávat s n hodnotami z. Právě tento fakt je důsledkem markantního nedostatku Wassersteinova autoencoderu. Porovnávání vzorků je výpočetně náročnou operací, proto je WAE několikrát pomalejší než VAE.

1. Pro 1000 náhodně generovaných hodnot z rozdělení $\mathcal{N}(5, 3)$ byl vytvořen model s počtem neuronů v síti `nx = nh = nz = 1`. Pro každou epochu byl vektor hodnot iterován 1000x a vnitřní vzorkování bylo také velikosti 1000 bodů. Z modelu bylo následně vygenerováno 1000 nových hodnot, které byly nafitovány jako normální rozdělení a porovnány s rozdělením původním. Grafické znázornění výsledku je na Obr. 2.9.
2. Dále se přesuneme do dimenze 2. Původní data pochází stejně jako u příkladu VAE z dvourozměrného normálního rozdělení se střední hodnotou $(3; 5)^T$ a kovariační maticí rovnou jednotkové matici. Struktura neuronové sítě byla zvýšena na počet neuronů `nx = nh = nz = 2`, vzorkování pro MMD vzdálenost bylo sníženo na 200. Z modelu bylo následně vygenerováno 500 nových hodnot. Na Obr. 2.10 jsou vidět nově vygenerovaná data z natrénovaného modelu po 200 epochách.

3. V dalším případě bylo rozdělení $\mathcal{N}([3;5],\mathbf{I})$ transformováno předpisem:

$$x_i' = 2x_i + y_i$$
$$y_i' = x_i + y_i.$$

Model měl stejnou strukturu jako v bodě 2. Výsledek po 20 epochách je na Obr. 2.11.

4. Nakonec bylo standardní gaussovo rozdělení $\mathcal{N}(\mathbf{0},\mathbf{I})$ ve 2D transformováno předpisem

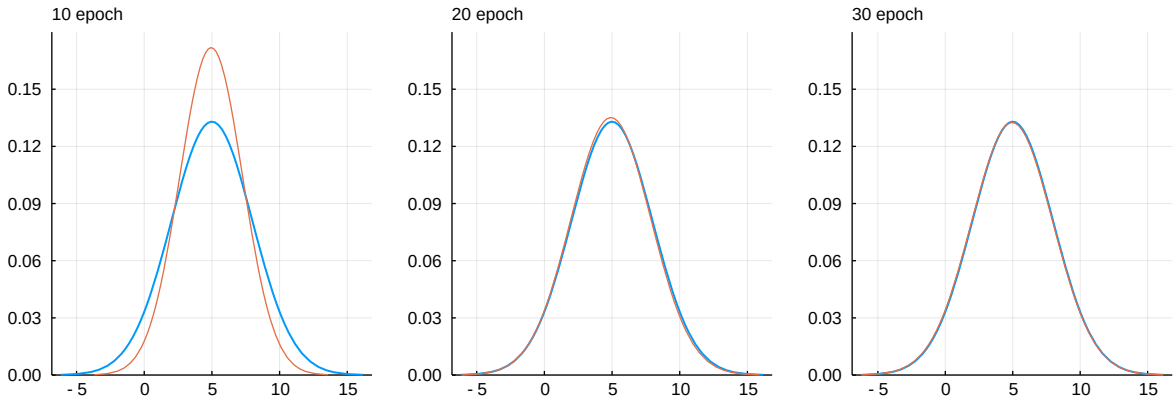
$$x_i' = \frac{1}{2}x_i + y_i^2$$
$$y_i' = 2y_i,$$

aby bylo možné ukázat, že Wassersteinův autoencoder lze naučit i komplexnější strukturu dat, kterou již nelze odhadovat klasickými modely. Model byl v tomto případě upraven následovně:

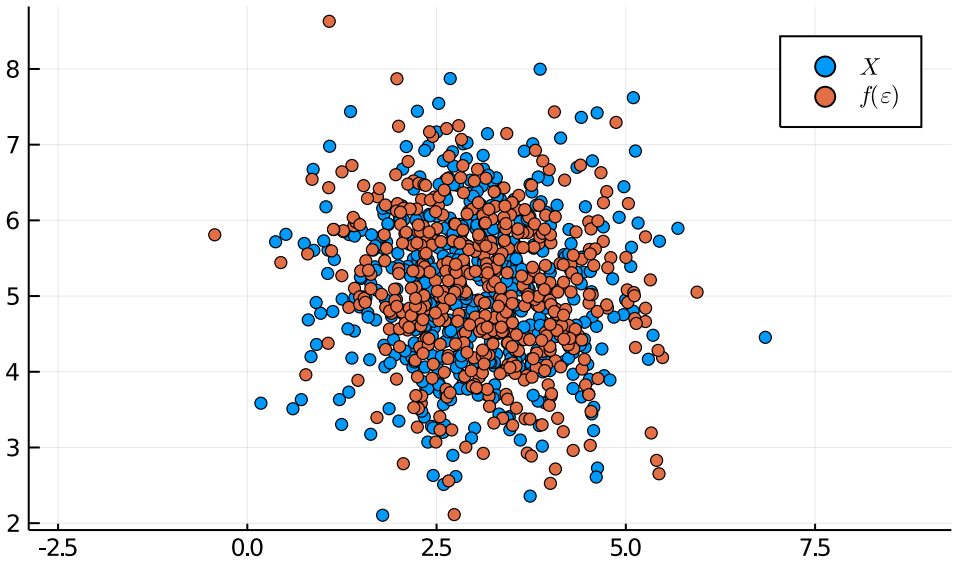
- nastavení počtu neuronů `nx = 2 = nz, nh = 4`,
- byla použita aktivační funkce swish v neuronové síti `A(x) = Dense(nx,nh,swish)` a `f(x) = Chain(Dense(nz,nh,swish),Dense(nh,nx))`,
- Gaussovský kernel byl nahrazen IMQ kernelem, který je definován v článku [12].

Výsledek je vidět na Obr. 2.12.

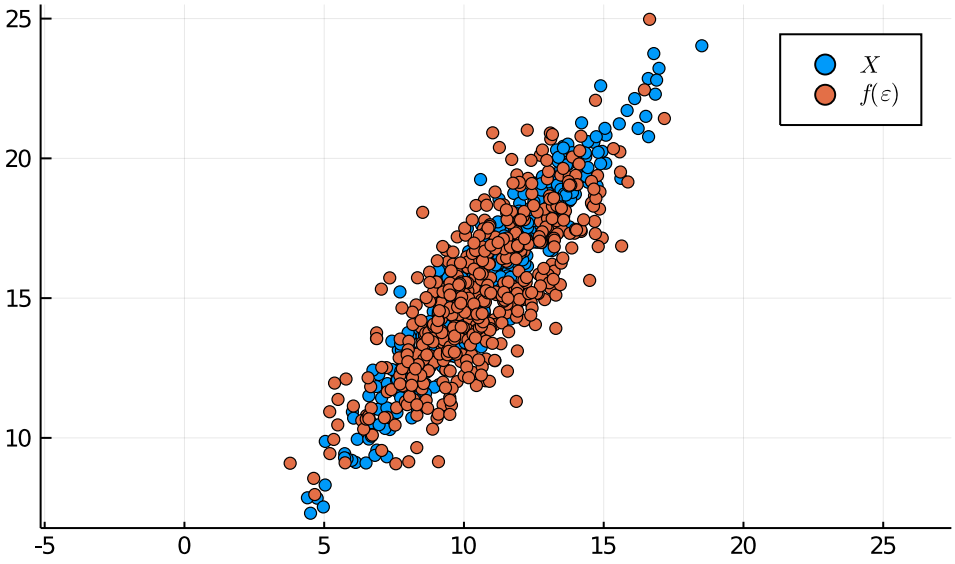
Výsledky Wassersteinova autoencoderu jsou podobné klasickému VAE. Nicméně trénování tohoto modelu je několikanásobně výpočetně náročnější. Na složitějších datech, viz. obrázek 2.11, je pak komplikovanější zajistit správný rozptyl modelu.



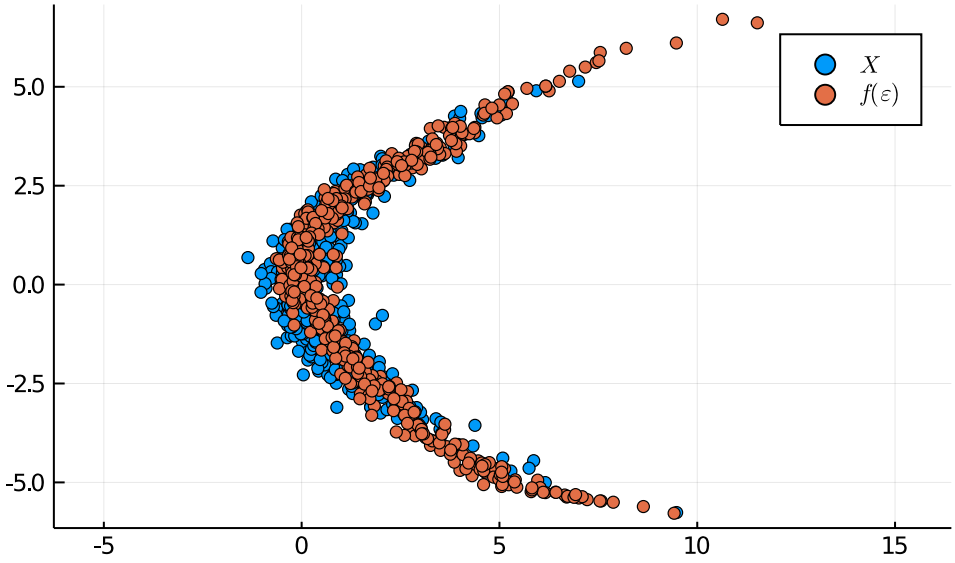
Obrázek 2.9: Porovnání pravé distribuce dat (modrá) s distribucí dat vygenerovaných z modelu WAE po daném počtu epoch.



Obrázek 2.10: Data vygenerovaná natrénovaným modelem WAE po 200 epochách pro data z bodu 2.



Obrázek 2.11: Data vygenerovaná natrénovaným modelem WAE po 200 epochách pro data z bodu 3.



Obrázek 2.12: Data vygenerovaná natrénovaným modelem WAE po 200 epochách pro data z bodu 4.

2.4 VampPrior

Variační autoencoder pracuje s nejjednodušším možným modelem apriorního rozdělení $p(Z)$, kterým je standardní gaussovo rozdělení. V rámci takového modelu může docházet k nevyužití některých latentních vrstev. V článku [14] autoři Jakub M. Tomczak a Max Welling proto zavádí nové apriorní rozdělení, které nazývají *variational mixture of posteriors* neboli variační směs aposteriorních rozdělení, zkráceně VampPrior.

Základní rovnici variačního autoencoderu (2.25) je možné přepsat způsobem

$$\mathcal{L} = \mathbb{E}_{q(Z|X)}\left[\ln p(X|Z)\right] + \mathbb{E}_{q(X)}\left[\mathbb{H}\left[q(Z|X)\right]\right] - \mathbb{E}_{q(Z)}\left[-\ln p(Z)\right], \tag{2.40}$$

čímž získáme dva regularizační členy namísto jednoho – střední hodnotu entropie aposteriorního rozdělení a cross-entropy mezi tzv. agregovaným aposteriorním rozdělením $q(Z) = \frac{1}{N}\sum_{n=1}^{N} q(Z|x_n)$ a apriornem.

Je třeba najít apriorní rozdělení, které nahradí původní standardní normální rozdělení $p(Z)$ z VAE a zároveň bude maximalizovat ELBO. Tímto rozdělením je právě agregované aposteriorní rozdělení

$$p^*(Z) = \frac{1}{N}\sum_{n=1}^{N} q(Z|x_n). \tag{2.41}$$

Pro velká N by však byla tato volba výpočetně náročná. Rozdělení $p^*(Z)$ lze však aproximovat směsí variačních aposteriorních rozdělení s pseudo-vstupy u_k:

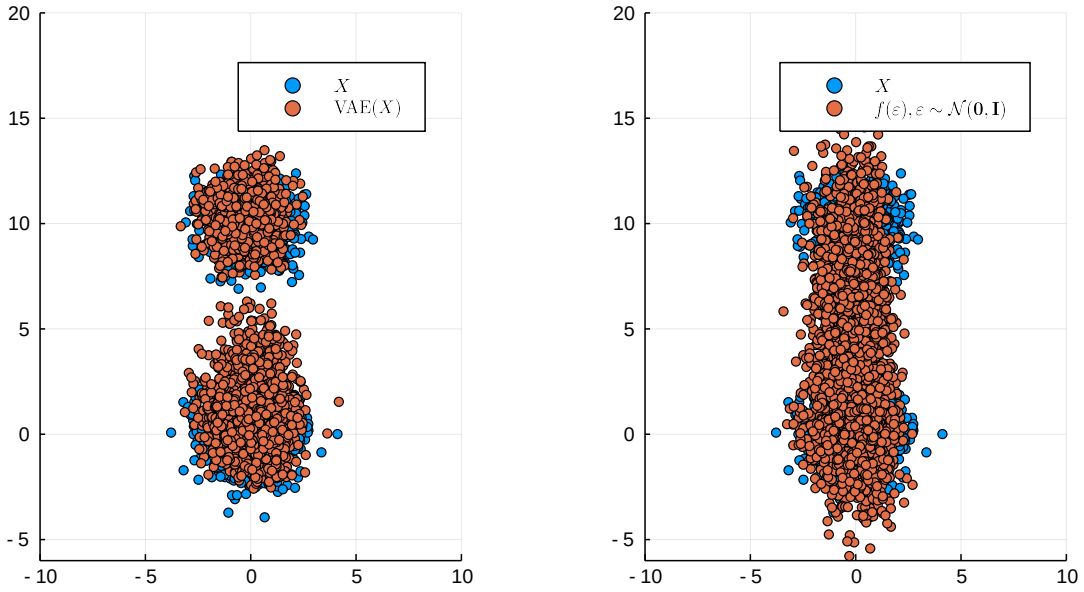
$$p(Z) = \frac{1}{K}\sum_{k=1}^{K} q(Z|u_k), \tag{2.42}$$

kde K je počet těchto pseudo-vstupů. Pseudo-vstupy je možné získat v rámci zpětné propagace a brát je jako hyperparametry apriorna. Získáváme tak multimodální apriorní rozdělení, které je bohatší než klasické $p(Z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, a tedy by mělo zabraňovat *over-regularization*, kdy aposteriorní rozdělení získává triviální formu standardního normálního rozdělení. Volbou $K \ll N$ lze zároveň snížit výpočetní náročnost.

2.5 Směs variačních autoencoderů

Jednou z dalších nevýhod klasického variačního autoencoderu je, že se snaží všechna data modelovat normálním rozdělením s diagonální kovarianční maticí a střední hodnotou jako funkcí dat. Tato skutečnost implikuje, že jednotlivé charakteristiky v latentním prostoru jsou modelovány nezávisle a unimodálně. Ve chvíli, kdy se jím pokusíme modelovat například směs gaussových rozdělení, bude mít problém.

Příklad můžeme vidět na Obr. 2.13. Byly vygenerovány dva vektory, jeden z rozdělení $\mathcal{N}(\mathbf{0}, \mathbf{I})$ o 2000 bodech a druhý z $\mathcal{N}([0; 10], \mathbf{I})$ o 1000 bodech. Variační autoencoder se zvládne naučit rekonstruovat data do příslušných clusterů, ale při generaci nových dat už má problém, jelikož nedokáže své rozdělení „roztrhnout“ na dvě oddělené části.



Obrázek 2.13: Data na levé straně představují vstupní data rekonstruovaná variačním autoencoderem. Levá strana ukazuje data nově generovaná modelem.

Wassersteinův autoencoder nabízí pro tento případ lepší řešení. Jak již bylo zmíněno dříve, volba $p(Z)$ může být libovolná. Pokud tedy zvolíme pro tento konkrétní příklad směs dvou normálních rozdělení, měli bychom dostat lepší výsledek než klasické VAE.

2.5.1 Motivace a inspirace

U Wassersteinova autoencoderu narážíme na problém, že se složitější volbou $p(Z)$ přidáváme další volitelné parametry modelu. V praxi například nevíme, z kolika směsí by bylo ideální $p(Z)$ poskládat. Navíc nemáme žádnou kontrolu nad tím, jakým způsobem je latentní prostor vytvářen a jak jsou přiřazovány jednotlivé body do příslušných clusterů.

Inspirací pro vytvoření nového modelu budou již dobře známé směsové metody. Při pravděpodobnostním odhadování distribucí se můžeme setkat s problémem, že jsou data multimodální. V takovém případě se vyplatí vytvořit směsový model. Dva hlavní postupy jsou následující:

1. Vytvořit pravděpodobnostní model, pro který bude platit $p(x_i|l_i) = \prod_k \mathcal{N}(\mu_k, \Sigma_k)^{l_{ik}}$, kde l_{ik} značí, do kterého rozdělení ze směsi patří bod x_i. Následně je možné definovat celý model a použít EM algoritmus pro získání odhadů parametrů rozdělení celé směsi.

2. Bayesovský přístup vychází ze stejné myšlenky, ale zavádí navíc apriorní rozdělení pro parametry μ_k a Σ_k.

Oba algoritmy dokáží dobře aproximovat data pocházející ze směsi normálních rozdělení. Bayesovský přístup má navíc tu výhodu, že není tolik závislý na původním počtu módů. Pokud data pocházejí z rozdělení se třemi módy, ale náš model definujeme s počáteční podmínkou, že módů je šest, pak správně nastavený variační bayesovský model dokonverguje k výsledku, kdy využije pouze tři ze svých komponent. Pravděpodobnost, že data pocházejí ze zbylých komponent, klesne na nulu.

2.5.2 EMVAE

V návaznosti na inspiraci předchozími modely se nabízí vytvořit model, který bude kombinovat již zmíněné myšlenky. Cílem je vytvořit model jako směs variačních autoencoderů. V první řadě tedy budeme potřebovat přiřadit data do příslušných clusterů (modelovaných variačními autoencodery), jež budou mít každý své vlastní rozdělení, a potom generovat z těchto modelů.

Problém rozšíříme tak, že přidáme novou proměnnou l_{ik}, která bude udávat pravděpodobnost, že datový bod s indexem i náleží do k-tého modelu, tedy ho kódujeme rozdělením $q_k(Z|X) = \mathcal{N}(\mu_k(X), \Sigma_k(X))$ a dekódujeme pomocí funkce $f_k(Z, \theta_k)$. Matematický zápis tak bude mít podobu

$$\begin{aligned} p(x_i, l_i) &= p(x_i|l_i)p(l_i) \\ p(x_i|l_i) &= \prod_{k=1}^{K} \mathcal{N}(x_i|f_k(z;\theta_k), \omega^2\mathbf{I})^{l_{ik}} \\ p(l_i = e_k) &= \alpha_k, \quad \text{kde} \quad \sum_{k=1}^{K} \alpha_k = 1. \end{aligned}$$

Jako základní model pro trénování použijeme známý EM algoritmus (*expectation maximization algorithm*) [15]. Z tohoto důvodu bude směsový model dále značen EMVAE.

Samotný EM algoritmus spočívá ve dvou krocích, které se iteračně opakují až do konvergence modelu:

1. *Expectation step*, ve kterém spočítáme střední hodnotu věrohodnostní funkce modelu v proměnné l, tj.

$$\begin{aligned} \ln p(x_1, l_1, \ldots, x_n, l_n|\theta) &\propto \sum_{i=1}^{n}\sum_{k=1}^{K} l_{ik}\left[\ln \mathcal{N}(x_i|f_k(z;\theta_k), \omega^2\mathbf{I}) + \ln\alpha_k\right] = \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} l_{ik}\left[-\frac{1}{2\omega^2}\left(x_i - f_k(Z = \mu_k(x) + \Sigma_k^{1/2}(X) * \varepsilon)\right)^2 + \ln\alpha_k\right], \\ \mathbb{E}_l\left[\ln p(x_1, l_1, \ldots, x_n, l_n|\theta)\right] &= \sum_{i=1}^{n}\sum_{k=1}^{K} \hat{l}_{ik}\left[-\frac{1}{2\omega^2}\left(x_i - f_k(Z = \mu_k(x) + \Sigma_k^{1/2}(X) * \varepsilon)\right)^2 + \ln\alpha_k\right] \end{aligned}$$

kde opět používáme známý reparametrizační trik, $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ a navíc

$$\hat{l}_{ik} = p(l_i = e_k|x_i, \theta^{(j)}) = \frac{\alpha_k \mathcal{N}(x_i|f_k(z,\theta_k), \omega^2\mathbf{I})}{\sum_{k=1}^{K}\alpha_k \mathcal{N}(x_i|f_k(z,\theta_k), \omega^2\mathbf{I})}. \tag{2.43}$$

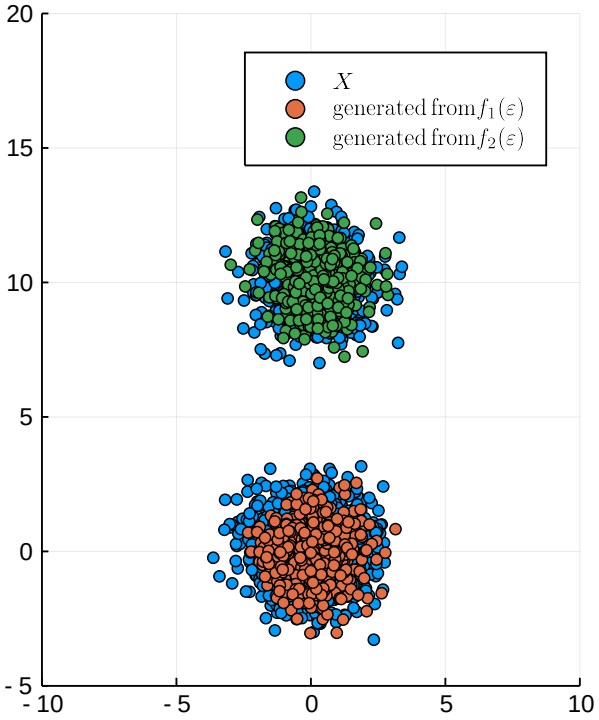
Tím spočítáme pravděpodobnostní vektor příslušnosti do jednotlivých clusterů pro každý bod x_i.

2. Následuje *maximization step*, ve kterém maximalizujeme vypočtenou střední hodnotu. V klasickém EM algoritmu maximalizujeme v hledaných proměnných a získáváme jejich bodové odhady. Zde však chceme maximalizovat v parametrech funkcí f_k. V tomto kroku se tedy vracíme k procesu trénování variačního autoencoderu a optimalizujeme ELBO. To dostává ve směsovém modelu podobu

$$\mathcal{L} = \sum_{k=1}^{K} \left[\hat{l}_{ik} \ln p \left(x_i | Z = \mu_k(x_i) + \Sigma_k^{1/2}(x_i) * \varepsilon \right) + \hat{l}_{ik} \mathcal{D}_{KL} \left[q_k(Z|x_i) || p(Z) \right] + \ln \alpha_k \right]. \tag{2.44}$$

Trénování modelu pak probíhá jednoduše. V každé iteraci jsou spočítány l_{ik} podle 2.43 pro všechny datové body a komponenty. Následně optimalizujeme negativní ELBO 2.44 jako ztrátovou funkci modelu.

Pro jednoduchý 2D příklad byl natrénován model ze směsi dvou autoencoderů. Výsledek pro stejná data jako výše je možné nahlédnout na Obr. 2.14.

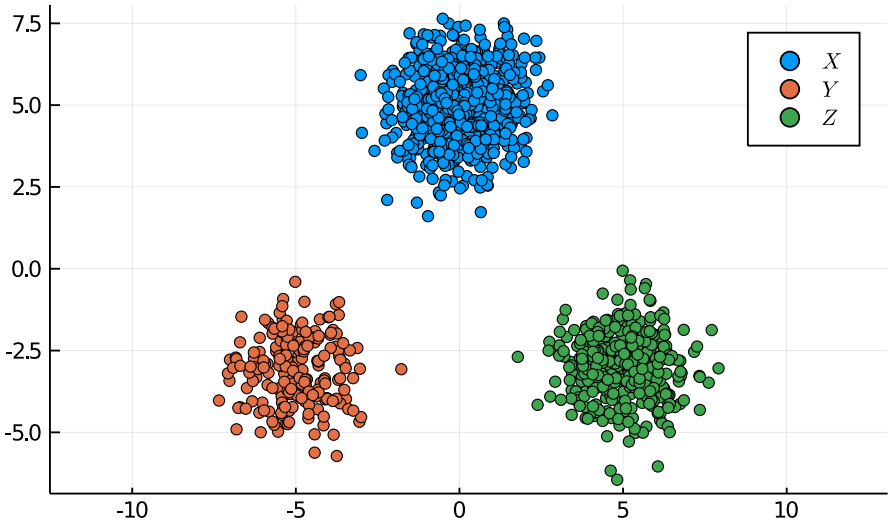


Obrázek 2.14: Původní a nově vygenerovaná data ze směsového modelu.

Model EMVAE si zachovává vlastnosti bayesovského směsového modelu. Vezměme si ilustrační příklad s daty generovanými ze tří dvourozměrných normálních rozdělení

$$\begin{aligned} X &\sim \mathcal{N}([0;5], \mathbf{I}), \ n_x = 800, \\ Y &\sim \mathcal{N}([-5;3], \mathbf{I}), \ n_y = 200, \\ Z &\sim \mathcal{N}([5;-3], \mathbf{I}), \ n_z = 400, \end{aligned} \tag{2.45}$$

kde n_x, n_y a n_z značí počet dat vygenerovaných z jednotlivých rozdělení. Všechna data spojíme do jedné matice, která bude představovat trénovací dataset. Vizualizace dat je na Obr. 2.15.



Obrázek 2.15: Data generovaná ze tří dvourozměrných normálních rozdělení podle (2.45).

Cílem bude vytvořit model, který dokáže generovat podobná data. Model definujeme jako směs šesti variačních autoencoderů, které budou mít všechny stejnou strukturu

```
nx = 2
nh = 2
nz = 2
A = Dense(nx,nh,swish)
mu = Dense(nh,nz)
log_sigma = Dense(nh,nz)
f = Chain(Dense(nz,nh,swish),Dense(nh,nx))
```

s volbou hyperparametru $\omega^2 = 0.3$.

Parametr α je zde vektorem, jehož složky určují pravděpodobnost, že daný pod pochází z k-té komponenty. Generování nových dat pak bude mít o něco složitější podobu než u klasického VAE:

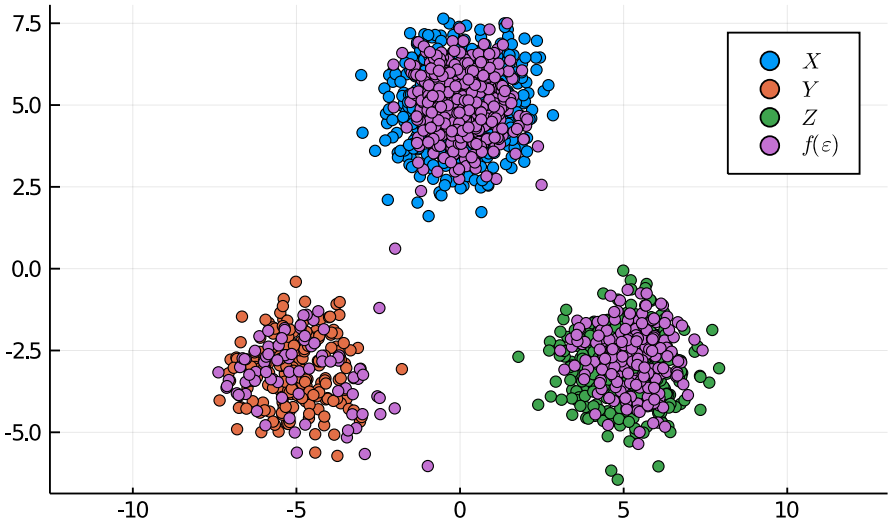
1. Vygenerujeme náhodně vektor $v \sim \text{Cat}(\alpha)$ z kategorického rozdělení s pravděpodobnostním vektorem α. Získáme tak vektor, pro který platí $v_i = 0 \; \forall i \neq k$ a $v_k = 1$.

2. Vygenerujeme vektor $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

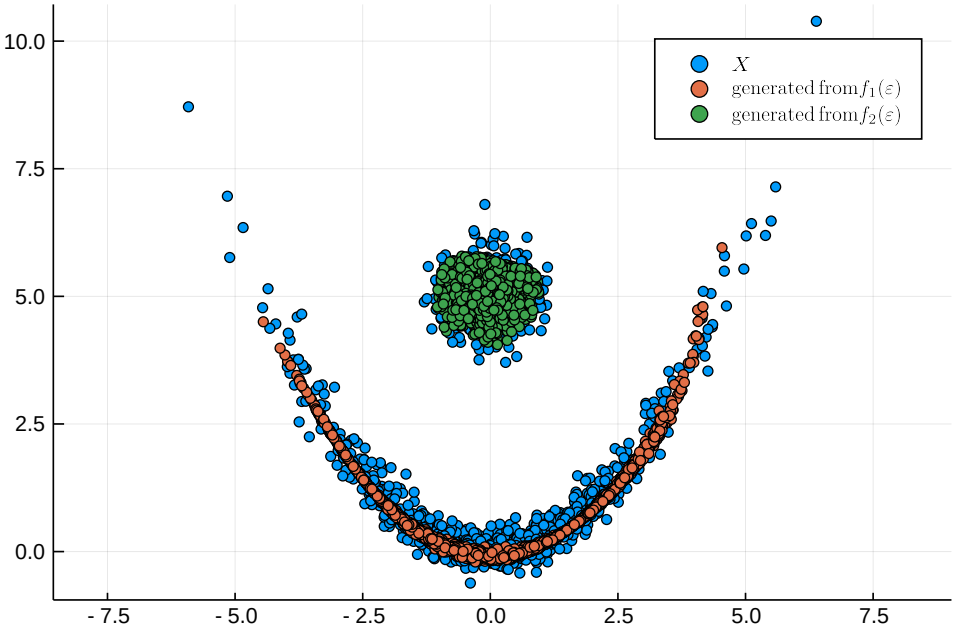
3. Vektor ε dekódujeme k-tým dekodérem f_k.

Po 200 epochách model dokonvergoval k vektoru $\alpha = (0{,}0;\ 0{,}57;\ 0{,}29;\ 0{,}0;\ 0{,}14;\ 0{,}0)$. Vidíme tedy jasně, že tři z komponent zůstaly nevyužity, jelikož rozdělení dat bylo možné namodelovat pouze třemi autoencodery. Budeme-li generovat data použitím algoritmu popsaného výše, získáme výsledek jako na Obr. 2.16. Lze si všimnout, že počet vygenerovaných dat v jednotlivých shlucích odpovídá rozdílným hodnotám α_k.

EMVAE lze následně použít na mnohem složitější struktury dat, jako například na Obr. 2.17.



Obrázek 2.16: Výsledek EMVAE pro data generovaná podle (2.45).



Obrázek 2.17: Směs variačních autoencoderů natrénovaná na složitějších datech.

Kapitola 3

Využití modelů v detekci anomálií

Modely popsané v předchozí kapitole lze využít na řadu reálných problémů. Jedním z nich je detekce anomálií. Algoritmy pro detekci anomálií jsou využívány například při odhalování podvodů ve finančních transakcích nebo v internetovém provozu pro detekci malwaru. Přístupy k tomuto problému mohou být různé, v rámci bakalářské práce se budeme zabývat využitím variačního autoencoderu na dva datasety z *UCI Machine Learning Repository* [16].

Jedním z možných způsobů, jak využít variační autoencoder, je sledovat rekonstrukční chybu, respektive rekonstrukční pravděpodobnost [17]. Model je natrénován na trénovacím datasetu, který tvoří pouze normální data. Následně je vypočítána rekonstrukční chyba natrénovaného modelu pro celý dataset obsahující i anomálie.

Vezmeme daný bod x a zjistíme hodnoty neuronových sítí μ a Σ. Následně náhodně vybereme L vzorků z z rozdělení $\mathcal{N}(\mu, \Sigma)$, které dekódujeme sítí f. Rekonstrukční pravděpodobnost je pak Monte Carlo odhadem $\mathbb{E}_{q(z|x)}\left[\log p(x|z)\right]$, tedy při vypuštění konstant

$$\text{rekonstrukční pravděpodobnost} = \frac{1}{L}\sum_{l=1}^{L}\left(x - f(z_l)\right)^2. \tag{3.1}$$

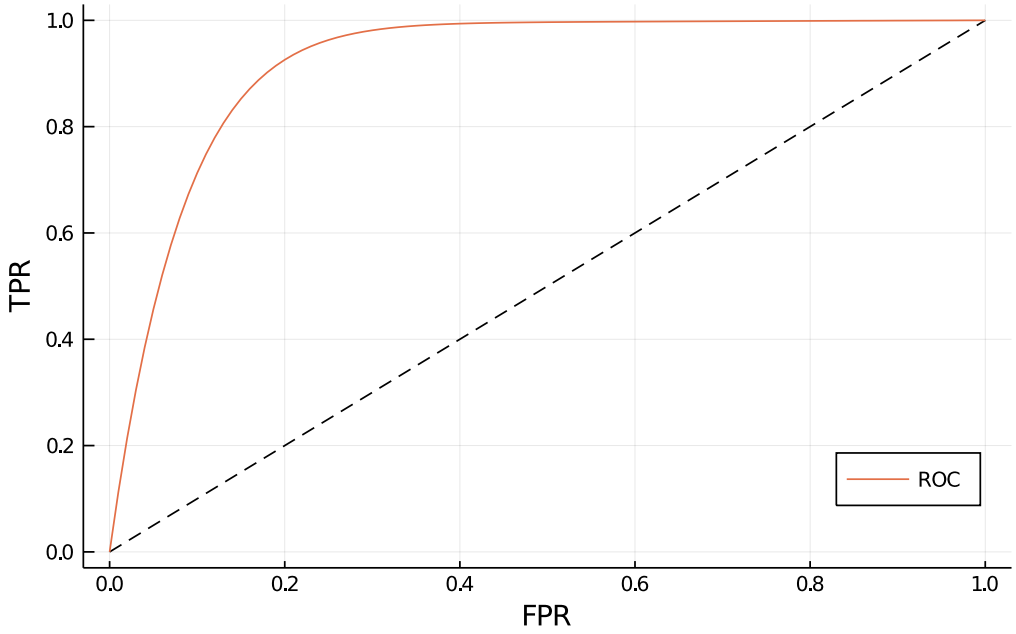
Při správně definovaném a natrénovaném modelu by pak anomální vzorky měly mít vyšší rekonstrukční chybu než normální, jelikož jejich rozdělení není stejné a variační autoencoder jej nezná.

3.1 Metriky pro měření kvality modelu

Aby bylo možné určit kvalitu modelu, je třeba zavést metriky, které se v oblasti detekce anomálií používají. Jednou z takových metrik je ROC křivka, neboli *receiver operating characteristics* [18]. ROC křivka vykresluje *true positive rate* (TPR) proti *false positive rate* (FPR). V binární klasifikaci se totiž snažíme balancovat dva problémy: chceme, abychom měli co nejméně falešně negativních případů, ale zároveň co nejméně případů falešně pozitivních. První z těchto chyb se nazývá chyba I. druhu a je zásadnější (nechceme například nemocného pacienta poslat domů, protože mu vyšel negativní test), než druhá z nich, chyba II. druhu (není tak katastrofální, pokud budeme léčit zdravého pacienta). ROC křivka ukazuje vztah mezi těmito dvěma charakteristikami.

Pro jednotlivé hodnoty TPR a FPR platí

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$
$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$



Obrázek 3.1: Příklad dobré ROC křivky, AUC = 0,92.

kde TP je počet správně klasifikovaných pozitivních vzorků, FP počet falešně pozitivních hodnot a TN počet správně klasifikovaných negativních vzorků.

V grafickém znázornění ROC křivky se často navíc vykresluje lineární křivka $y = x$, která představuje hranici, na které jsou anomální a normální vzorky určovány náhodně s 50% úspěšností. Pokud se tedy výsledná ROC křivka pohybuje nad touto hranicí, dává lepší výsledky než náhodný výběr.

Jako další metriku pro určení kvality modelu lze použít plochu pod ROC křivkou, která bude dále označena jako AUC (z anglického *area under curve*). Příklad dobré ROC křivky je vidět na Obr. 3.1. Pro tuto křivku je hodnota AUC = 0,92.

3.2 Ilustrační kód a popis jednotlivých modelů

Pro všechny experimenty v následujících částech bude použita stejná struktura modelu neuronových sítí s rozdílem v počtu parametrů `nx, nh, nz` v podobě

```
A = Dense(nx,nh,sigmoid)
mu = Dense(nh,nz)
logs = Dense(nh,nz)
f = Chain(Dense(nz,nh,sigmoid),Dense(nh,nx))
ps = Flux.params(A, mu, logs, f).
```

Reparametrizační trik je pak realizován funkcí `z`, KL divergence je definovaná podle (2.31):

```
z(mu, logs) = mu .+ exp.(logs) .* randn(size(mu))
KL(mu, logs) = 0.5 * sum((exp.(logs)).^2 .+ mu.^2 .- 1.0 .- 2. * logs).
```

Jako optimalizační nástroj byl použit ADAM s rozdílným nastavením *learning rate* η.

Trénování jednotlivých modelů je pak realizováno funkcí `Flux.train!`. Ztrátová funkce pro klasický variační autoencoder má podobu

```
function loss(x)
  HID = A(x);
  zsample = z(mu(HID), logs(HID))
  0.5 * sum((x .- f(zsample)).^2) + s * KL(mu(HID), logs(HID))
end.
```

Ztrátová funkce pro model Wassersteinova autoencoderu je pak definovaná následovně

```
function loss(x)
  HID = A(x);
  zsample = z(mu(HID), logs(HID))
  sample_rnd = hcat([randn(nz) for i in 1:n_sample]...)
  sample_z = hcat([z(mu(HID), logs(HID)) for i in 1:n_sample]...)
  0.5 * sum((x .- f(zsample)).^2) + s * mmd(kernel, sample_rnd, sample_z)
end,
```

kde `n_sample` je počet generovaných vzorků z rozdělení $p(Z)$.

Pro modely Wassersteinova autoencoderu byly vybrané dva kernely: gaussovský s šířkou γ a inverzní multikvadratický kernel (dále označován jako IMQ kernel) definovaný jako $k(d) = C/(C+d)$, kde d je vzdálenost podle [12]. Oba kernely i MMD vzdálenost jsou implementovány pomocí balíčku IPMeasures.jl. Pro Wassersteinův autoencoder a následující experimenty bylo zvoleno jednoduché $p(Z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Značení modelů

Pro oba datasety byly pro každý model (VAE, WAE, EMVAE, ...) použity dvě architektury sítě rozdělené podle počtu neuronů. Pro přehlednost je proto zavedeno následující značení:

- typ modelu (VAE, WAE, ...) je značen dolním indexem,
- složitost architektury je značena horním indexem, který představuje počet `nh` neuronů ve skryté vrstvě,
- u Wassersteinova autoencoderu je kernel rozlišen jako (G) a (IMQ) pro gaussovský a IMQ kernel.

Jednoduchý model klasického VAE s počtem neuronů v síti `nz = nh = 2` tak bude označen jako $\mathrm{M}^{(2)}_{\mathrm{VAE}}$.

3.3 KDD Cup 1999

KDD Cup 1999 dataset je soubor dat využívaný jako ilustrativní problém v klasifikaci anomálií. Úkolem je vytvořit model, který dokáže rozlišit mezi „dobrým“ a „špatným“ připojením, tedy detekovat možné narušitele.

Celkem se v datech nachází 41 příznaků, z nichž 34 je spojitých a 7 kategorických. Rozdělují se do tří hlavních kategorií. První jsou základní charakteristiky jednotlivých spojení jako délka spojení nebo objem přenesených dat. Druhá skupina jsou příznaky související s daným spojením jako například počet nevydařených pokusů o přihlášení nebo jestli byl uživatel přihlášen či ne. Poslední kategorie udává příznaky vypočítané v rámci dvousekundového intervalu připojení.

V následujícím příkladu bude použit menší počet těchto příznaků. Konkrétně budeme používat pouze připojení s protokolem http a všechny spojité příznaky. Zde je nutné podotknout, že v definici datasetu je jako spojitý příznak uvedený i počet zamítnutých přihlášení, který dosahuje hodnot 1–4. Podobné příznaky nebyly použity pro trénování modelu. Použit byl dataset obsahující pouze 10 % trénovacích dat, ve kterém se nachází 61886 normálních a 2407 anomálních vzorků.

Trénování probíhalo pouze na normálních datech. Následně byla spočítána rekonstrukční pravděpodobnost podle (3.1) a příslušná ROC křivka spolu s AUC pro daný model.

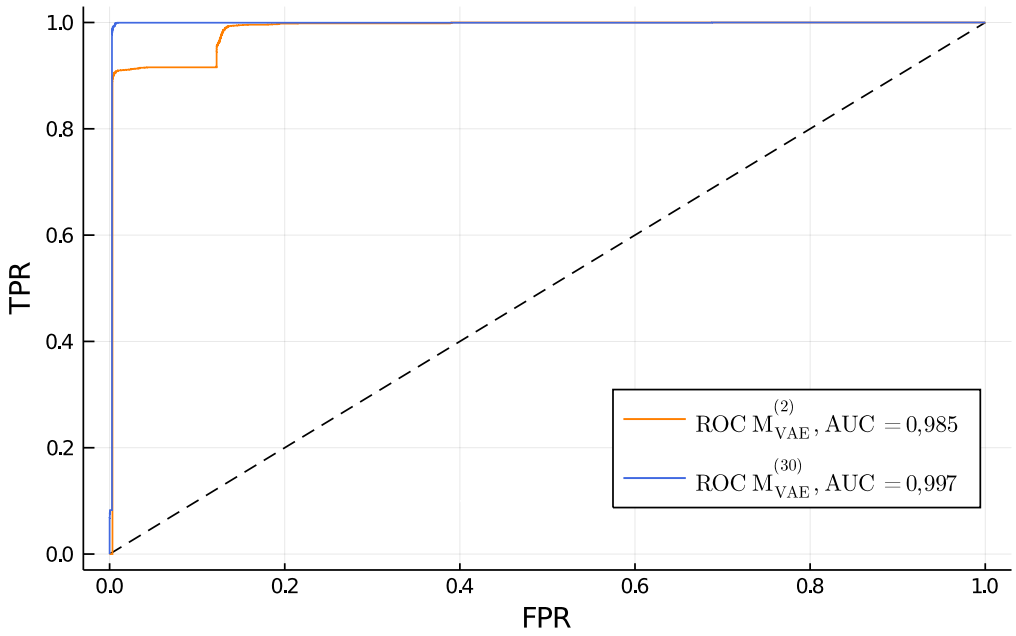
Zkoumané modely

Pro KDD Cup dataset byly porovnávány modely VAE, WAE (zvlášť pro gaussovský kernel a IMQ kernel) a EMVAE. Pro modely VAE a WAE byly zkoumány dvě hlavní architektury rozdělené podle počtu neuronů v sítích `A, mu, logs`. Pro jednodušší model se jednalo o počty neuronů `nz = nh = 2`, pro složitější strukturu pak `nz = 10, nh = 30`.

3.3.1 VAE

Variační autoencoder není výpočetně tak náročný model jako WAE nebo EMVAE. Z tohoto důvodu byl model natrénován několikrát. Obecně se totiž jedná o stochastické modely s náhodnou inicializací parametrů sítě, jednotlivé výsledky se tak mohou lišit. Výsledné hodnoty AUC jsou pak průměrem z šesti běhů pro jednotlivé modely. Nejlepšího výsledku bylo dosaženo pro hodnotu rozptylu $\omega^2 = 300$.

Model $\mathrm{M}_{\mathrm{VAE}}^{(30)}$ dosáhl vysoké přesnosti s hodnotou AUC $= 0{,}997$. Jednodušší model $\mathrm{M}_{\mathrm{VAE}}^{(2)}$ dosáhl výsledku AUC $= 0{,}985$. Srovnání ROC křivek pro oba modely je na Obr. 3.2.



Obrázek 3.2: ROC křivky pro dataset KDD Cup 1999 a modely $\mathrm{M}_{\mathrm{VAE}}^{(2)}$ i $\mathrm{M}_{\mathrm{VAE}}^{(30)}$.

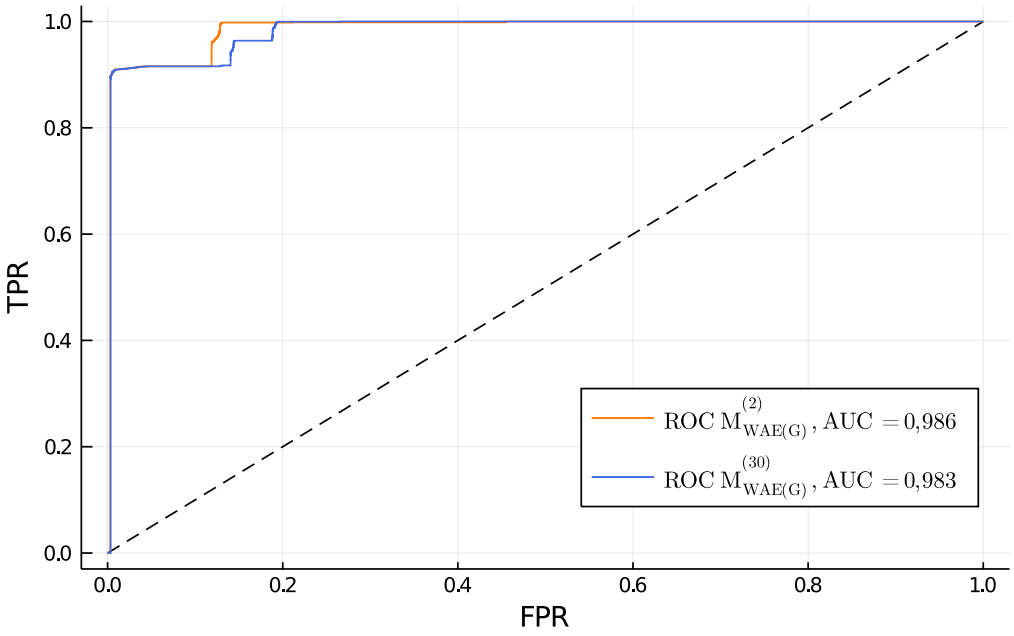
3.3.2 WAE

U Wassersteinova autoencoderu se ukázaly nepatrně větší rozdíly mezi jednoduššími a složitějšími modely, nicméně v opačném duchu než by se dalo očekávat. Lepších výsledků dosáhly pro gaussovský i IMQ kernel jednodušší modely s počty neuronů `nz = nh = 2`. Model $\mathrm{M}^{(2)}_{\mathrm{WAE(G)}}$ s nastavením parametrů $\gamma = 0{,}01$ a $\lambda = 1$ dosáhl hodnoty AUC $= 0{,}986$ a model $\mathrm{M}^{(30)}_{\mathrm{WAE(G)}}$ pak AUC $= 0{,}983$. Pro modely s využitím IMQ kernelu a nastavením hyperparametrů $c = 1$ a $\lambda = 1$ pak model $\mathrm{M}^{(2)}_{\mathrm{WAE(IMQ)}}$ skončil s hodnotou AUC $= 0{,}985$ a model $\mathrm{M}^{(30)}_{\mathrm{WAE(IMQ)}}$ na hodnotě AUC $= 0{,}982$. Výsledné ROC křivky pro gaussovský kernel jsou na Obr. 3.3. Výsledky pro IMQ kernel pak na Obr. 3.4.

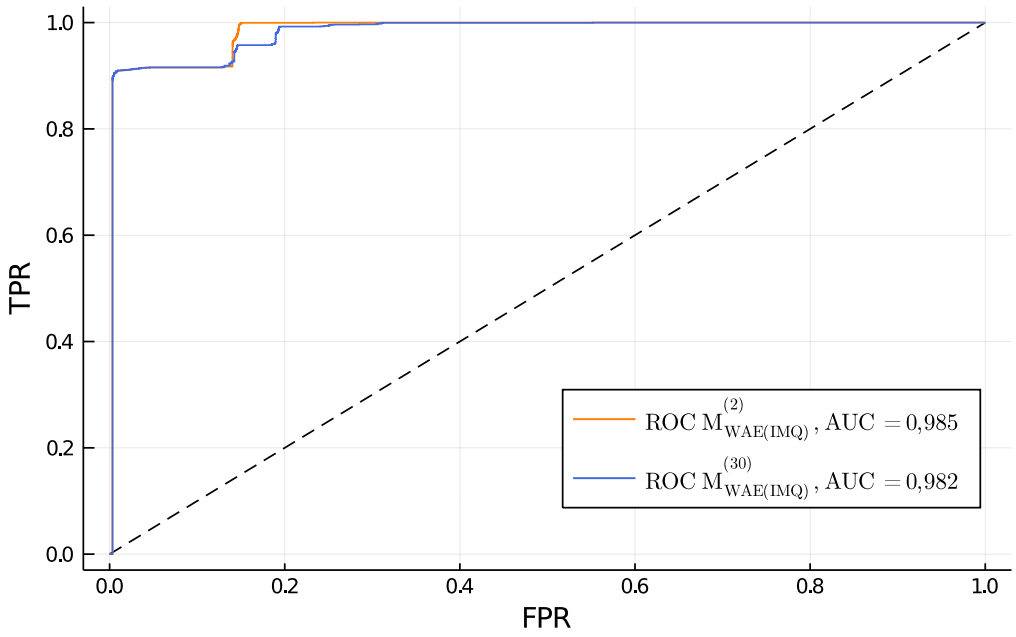
3.3.3 EMVAE

Na KDD Cup dataset byl použit i model EMVAE. Model byl definován pro 5 komponent stejných variačních autoencoderů s počty neuronů `nz = 10, nh = 30`. Při nastavení dostatečně vysokého rozptylu modelu $\omega^2 = 300$ bylo dosaženo toho, aby se model učil na více než jedné komponentě, konkrétně na dvou. Přestože trénování probíhalo na dvou komponentách, počet bodů příslušných do konkrétních clusterů se skokově měnil v průběhu jednotlivých epoch. Přesto však model dokázal dosáhnout vysoké přesnosti, konkrétně AUC $= 0{,}997$. Příklad ROC křivky je vidět na Obr. 3.5.

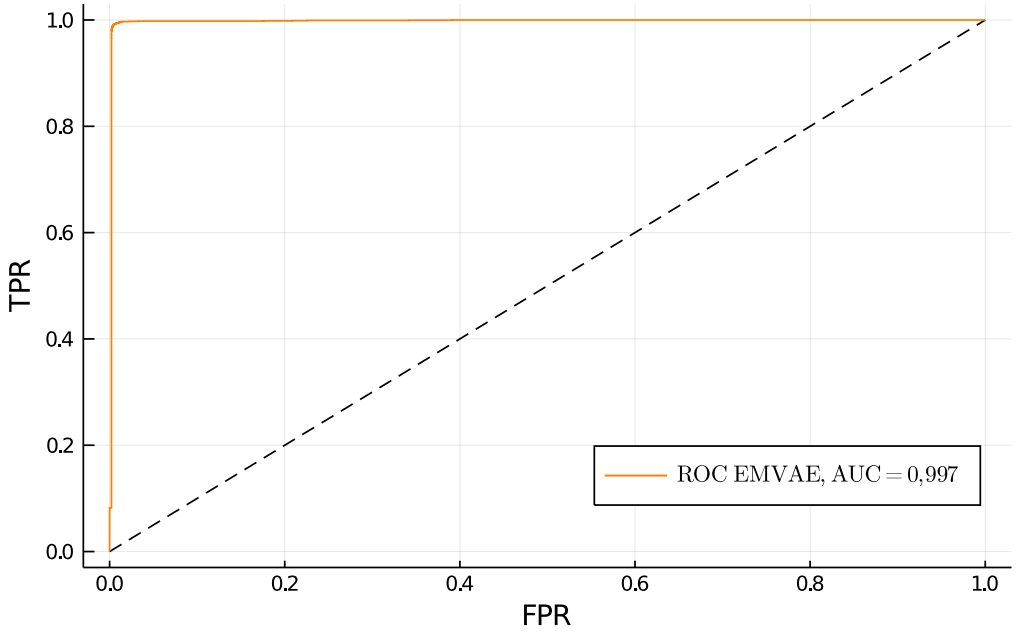
Vyzkoušen byl i jednodušší model s `nz = nh = 2`. Takový model nedokázal ani pro nastavení vysokého rozptylu rozdělit učení na více komponent.



Obrázek 3.3: ROC křivky pro dataset KDD Cup 1999 a modely $\mathrm{M}^{(2)}_{\mathrm{WAE(G)}}$ i $\mathrm{M}^{(30)}_{\mathrm{WAE(G)}}$.



Obrázek 3.4: ROC křivky pro dataset KDD Cup 1999 a modely $M^{(2)}_{WAE(IMQ)}$ i $M^{(30)}_{WAE(IMQ)}$.



Obrázek 3.5: ROC křivka pro dataset KDD Cup 1999 a model EMVAE.

3.3.4 Výsledky

Nejlepšího výsledku pro KDD Cup dataset dosáhl model variačního autoencoderu se sítí `nz = 10, nh = 30` a hodnotou hyperparametru $\omega^2 = 300$. Bylo dosaženo hodnoty AUC = 0,997. Stejného výsledku dosáhl i model EMVAE, který se skládal z pěti komponent tohoto variačního autoencoderu. Pro zbylé modely se neprokázaly markantní rozdíly ve výsledcích, všechny se pohybovaly kolem stejné hodnoty AUC = 0,985. ROC křivky měly pro tyto modely podobný průběh, kdy i při malém FPR bylo dosaženo vysoké hodnoty TPR kolem 0,9. Nicméně hodnota FPR musela být následně zvýšena, aby skokově došlo ke zvýšení TPR na hodnotu 1. Tato skutečnost ukazuje, že v datech se nejspíše nachází skupina anomálií, kterou modely popisovaly stejně jako určitou skupinu normálních dat. Pouze VAE a EMVAE dokázaly tuto strukturu v datech rozeznat a dosáhnout tak větší přesnosti.

Klasický variační autoencoder se pro tento dataset ukázal jako nejvhodnější model, jelikož dosáhl vysoké přesnosti za mnohem kratší čas než zbylé modely.

3.4 HTRU2

Dataset HTRU2 [19] obsahuje celkem 17898 vzorků a cílem je mezi těmito kandidáty detekovat pulsary. Pulsar je druhem neutronové hvězdy a jeho záření je detekováno rádiovými teleskopy. Jedná se o rotující tělesa, tedy detekované záření má určitou periodicitu, která je zaznamenána v průběhu času. Velmi často je však detekované záření způsobeno pouze šumem a vzájemnou interferencí rádiových vln. Proto se nabízí zkoumat modely strojového učení, které by dokázaly rozlišit mezi šumem a detekcí pravého pulsaru.

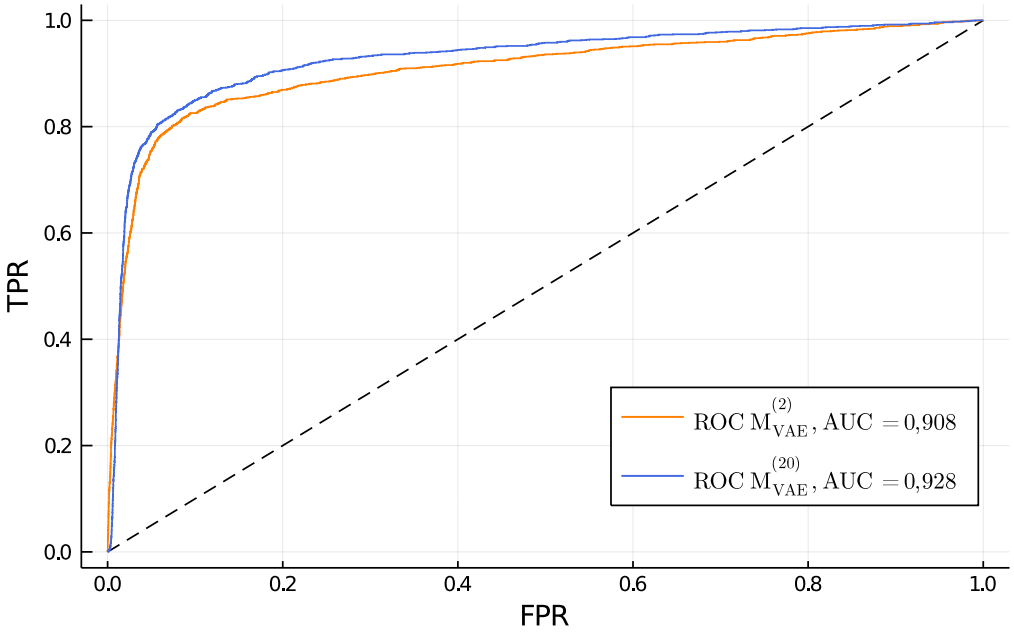
Dataset obsahuje 8 příznaků. Celkem se zde nachází 1639 pozitivních vzorků, které byly určeny jako pulsary. Trénování probíhalo opět pouze na normálních datech (tedy negativních vzorcích).

Zkoumané modely

Pro HTRU2 dataset byly porovnávány modely VAE, WAE (zvlášť pro gaussovský kernel a IMQ kernel) a EMVAE. Pro modely VAE a WAE byly zkoumány dvě hlavní architektury rozdělené podle počtu neuronů v sítích `A, mu, logs`. Pro jednodušší model se jednalo o počty neuronů `nz = nh = 2`, pro složitější strukturu pak `nz = 4, nh = 20`.

3.4.1 VAE

Bylo spuštěno několik modelů variačního autoencoderu pro jednodušší i složitější architekturu sítě. Pro HTRU2 dataset se projevily rozdíly mezi těmito dvěma architekturami, jak se dalo očekávat. Zatímco model $M_{VAE}^{(20)}$ dosáhl průměrné AUC = 0,928. Model $M_{VAE}^{(2)}$ skončil s hodnotou AUC = 0,908. Hodnoty AUC jsou průměrem z šesti běhů jednotlivých modelů. Porovnání ROC křivek pro oba modely je na Obr. 3.6. Nastavení rozptylu bylo $\omega^2 = 1$.



Obrázek 3.6: ROC křivky pro dataset HTRU2 a modely $M^{(2)}_{VAE}$ i $M^{(20)}_{VAE}$.

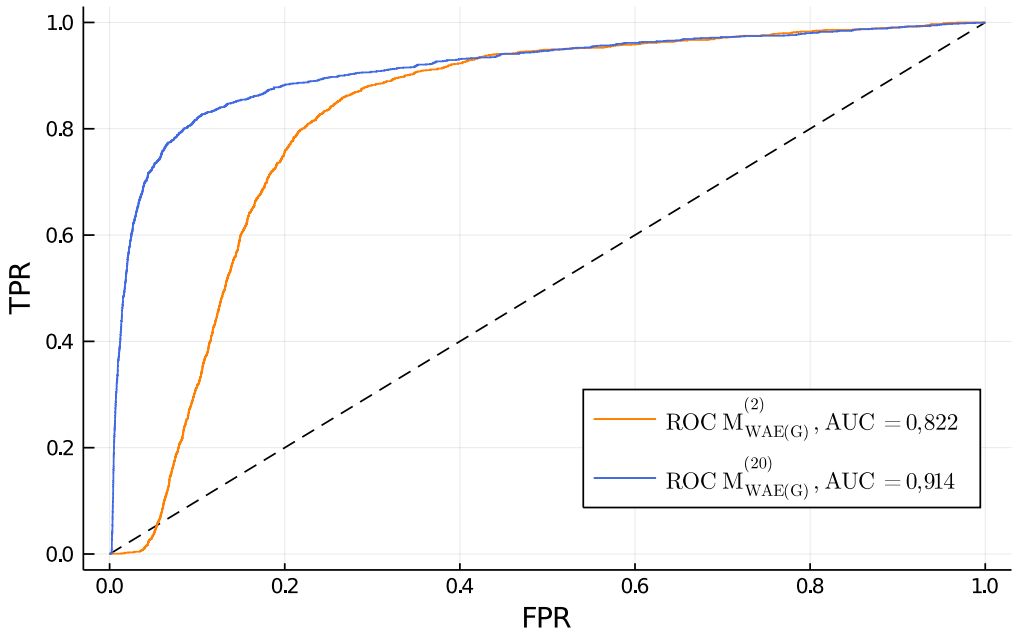
3.4.2 WAE

Wassersteinův autoencoder pro oba kernely ukázal podobné chování jako VAE. Model WAE s gaussovským kernelem $M^{(2)}_{WAE(G)}$ dosáhl hodnoty AUC pouze 0,822, což je nejhorší výsledek ze všech modelů. Naopak model $M^{(20)}_{WAE(G)}$ ukázal lepší přesnost s AUC $= 0{,}914$, což se více blíží výsledku $M^{(20)}_{VAE}$, ale nedosahuje ho. Porovnání obou ROC křivek je na Obr. 3.7. Hyperparametry pro oba modely byly $\gamma = 1$, $\lambda = 1$.

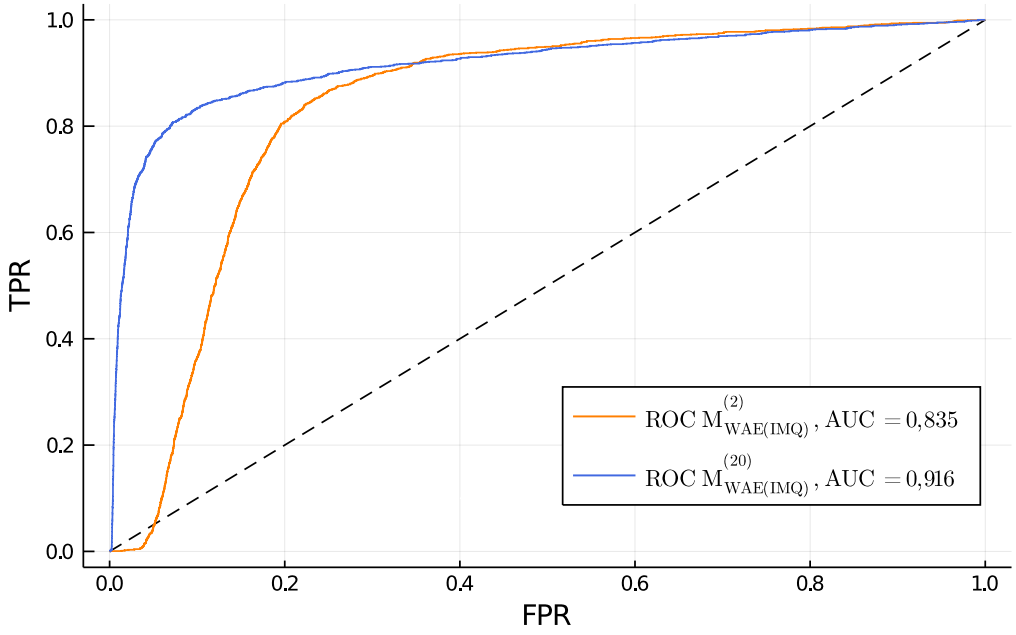
Na modelu s IMQ kernelem lze vidět podobné výsledky. Model $M^{(2)}_{WAE(IMQ)}$ dosáhl pouze AUC $= 0{,}835$, nicméně složitější model $M^{(20)}_{WAE(IMQ)}$ se dostal na hodnotu AUC $= 0{,}916$, což je srovnatelné s modelem $M^{(20)}_{WAE(G)}$. ROC křivky je možné nahlédnout na Obr. 3.8. Nastavení hyperparametrů bylo $c = 1$, $\lambda = 1$.

3.4.3 EMVAE

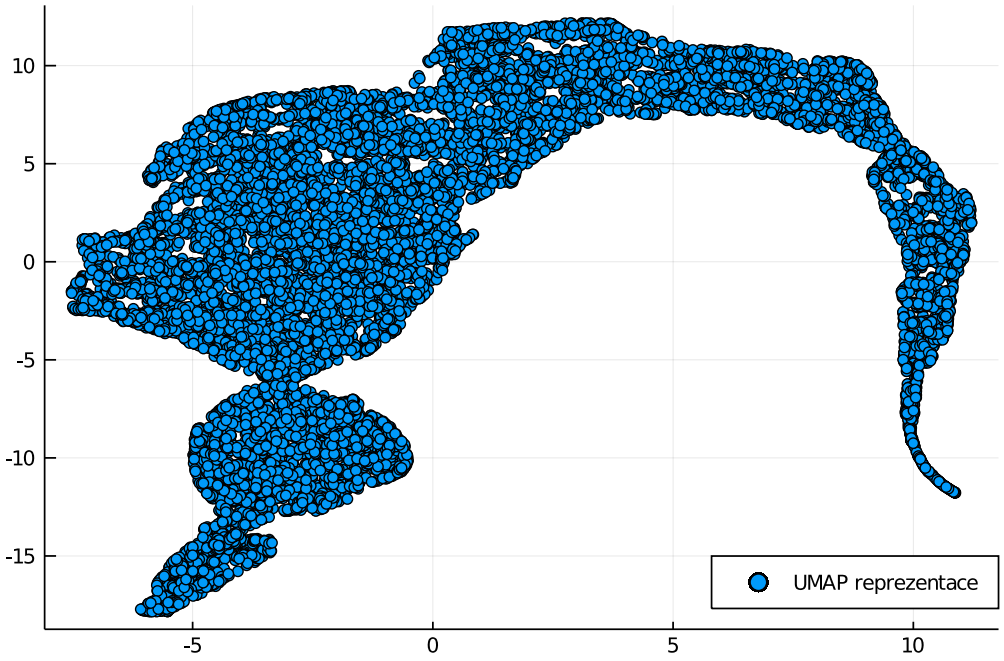
Model směsi variačních autoencoderů se bohužel na dataset HTRU2 neukázal jako vhodný. Při všech nastaveních hyperparametrů model dokonvergoval k učení se pouze na jedné komponentě, čímž se prakticky stal klasickým variačním autoencoderem. Model tak nedokázal rozlišit více separovatelných shluků v datech. Tato skutečnost je pochopitelná, podíváme-li se na projekci normálních dat z HTRU2 datasetu do dvou dimenzí pomocí UMAP [20] na Obr. 3.9. Jak je vidět, v datech nejsou jasně viditelné odlišitelné struktury, které by se mohly naučit každá jiným autoencoderem a tím zaručit lepší rekonstrukci.



Obrázek 3.7: ROC křivky pro dataset HTRU2 modely $M^{(2)}_{WAE(G)}$ i $M^{(20)}_{WAE(G)}$.



Obrázek 3.8: ROC křivky pro dataset HTRU2 modely $M^{(2)}_{WAE(IMQ)}$ i $M^{(20)}_{WAE(IMQ)}$.



Obrázek 3.9: 2D reprezentace normálních dat pro dataset HTRU2 pomocí UMAP.

3.4.4 Výsledky

Jako nejlepší model se ukázal klasický variační autoencoder s nastavením počtu neuronů v síti `nz = 4, nh = 20`. Pro dataset HTRU2 se ukázalo, že složitější modely s vyšším počtem neuronů dokázaly lépe popsat normální data a dosáhly vyšší hodnoty AUC a tedy úspěšnosti v klasifikaci anomálií. Model EMVAE v těchto datech nedokázal rozlišit oddělené shluky a učit se na více komponentách, tedy nepřináší lepší výsledky než klasické VAE.

Závěr

Cílem bakalářské práce bylo seznámit se s modelem variačního autoencoderu a jeho rozšířeními a nastudované modely dále použít na jednoduchá ilustrační i reálná data.

V první kapitole byly zavedeny základní pojmy a dále vysvětleny bayesovské principy na příkladu bayesovské regrese. Pro pozdější potřebu byly popsány neuronové sítě. Byl představen algoritmus backpropagation a různé optimalizační metody jako gradientní sestup nebo pokročilejší ADAM. Zmíněny byly i různé aktivační funkce spolu s výhodami i nevýhodami. Krátce byl představen i programovací jazyk Julia a balíček Flux.jl, který je v práci využit pro modelování neuronových sítí.

V druhé kapitole byly nejprve odvozeny a na příkladu demonstrovány ELBO odhady parametrů. Na tyto poznatky navázalo odvození základní rovnice variačního autoencoderu a celková definice tohoto modelu. Variační autoencoder byl následně použit jako generativní model na jednoduchých příkladech. Dále byl představen model Wassersteinova autoencoderu, který umožňuje složitější volbu rozdělení latentních proměnných. Model byl následně použit na stejná data jako klasický autoencoder. Krátce byl zmíněn i model s použitím VampPrior.

Ukázalo se, že variační autoencoder není vhodným modelem pro popis dat, která pocházejí z několika různých separovatelných rozdělení. S využitím inspirace směsovým bayesovským modelem tak byl sestaven nový model směsi variačních autoencoderů s použitím EM algoritmu, dále označovaný jako EMVAE. Na jednoduchých příkladech byla demonstrována jeho výhoda oproti klasickému variačnímu autoencoderu.

Ve třetí kapitole byly definované modely použity na reálná data v oblasti detekce anomálií. Pro možnost porovnání modelu byly zavedeny metriky jako ROC křivka a plocha pod touto křivkou AUC.

Prvním zkoumaným datasetem byl KDD Cup 1999. Nejlepšího výsledku dosáhly modely klasického VAE a EMVAE s počty neuronů `nz = 10, nh = 30`, konkrétně se jednalo o hodnotu AUC $= 0{,}997$. Ostatní modely dosáhly velmi podobných výsledků s hodnotou AUC $\sim 0{,}985$.

Druhý dataset, HTRU2 pro detekci pulsarů, přinesl větší rozdíly mezi jednotlivými modely. Nejlepšího výsledku dosáhl klasický variační autoencoder s počtem neuronů v síti `nz = 4, nh = 20`. Obecně se ukázalo, že pro HTRU2 data složitější modely dosáhly větší přesnosti, jelikož dokázaly lépe popsat normální data. Model EMVAE nedokázal v datech rozeznat více shluků dat a učit se na více komponentách, tato skutečnost má však své opodstatnění ve vlastní struktuře dat.

V návaznosti na získané poznatky se nabízí pokračovat ve zkoumání zavedených modelů, případně jejich vylepšení. Pro model EMVAE by bylo možné nahradit EM algoritmus jiným postupem, který by byl rychlejší a stabilnější.

Literatura

[1] Philip Protter Jean Jacod. *Probability Essentials*. Springer-Verlag Berlin Heidelberg, 2004.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[5] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[6] Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah. Fashionable modelling with Flux. *CoRR*, abs/1811.01457, 2018.

[7] Xitong Yang. Understanding the variational lower bound, 2017.

[8] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[9] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[10] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[11] Petersen, Michael Syskind Pedersen, Jan Larsen, Korbinian Strimmer, Lars Christiansen, Kai Hansen, Liguo He, Loic Thibaut, Miguel Barão, Stephan Hattinger, Vasile Sima, and We The. The matrix cookbook.

[12] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

[13] A Gretton, K. Borgwardt, Malte Rasch, B. Schölkopf, and AJ Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13:723–773, 03 2012.

[14] Max Welling Jakub M. Tomczak. VAE with a VampPrior. *arXiv preprint arXiv:1705.07120*, 2017.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[16] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[17] Jinwon An and Sungzoon Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[18] Tom Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[19] Robert J Lyon, BW Stappers, Sally Cooper, JM Brooke, and JD Knowles. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123, 2016.

[20] Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Seznam obrázků