

KMI/UMIN Umělá inteligence

Deep Learning

3. Úlohy, reprezentace dat a ztrátové funkce

Jan Konečný

Katedra informatiky, PřF UP v Olomouci

5. října 2026

Co nás dnes čeká

Klasifikace a regrese

Co síť dokáže reprezentovat

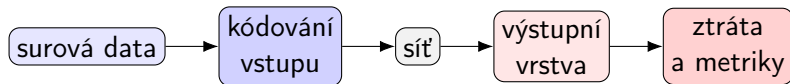
Reprezentace vstupů

Výstupní vrstva a ztráta

Aktivační funkce skrytých vrstev

O co pude

Typ úlohy určuje podobu výstupní vrstvy a s ní i volbu ztrátové funkce. Na druhém konci sítě je třeba rozhodnout, jak data vůbec zapsat jako čísla.



Půjdeme touto cestou zprava doleva: nejdřív úloha a metriky, pak výstup a ztráta, nakonec vstupy a skryté vrstvy.

Skripta: kap. 3 *Úlohy neuronových sítí a reprezentace dat*

Dvě základní úlohy

Klasifikace

Přirazení vstupu do jedné z předem daných tříd. Hledáme klasifikátor, který minimalizuje **chybu zobecnění**

$$E = \sum_{(x,y)} D(x,y) L(y, Cl(x))$$

s nula-jedničkovou ztrátou L .

Regrese

Predikce spojitě hodnoty. Hledáme f minimalizující rozdíl mezi predikcí a cílem, typicky

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2.$$

Skutečnou chybu zobecnění neznáme

Rozdělení D nemáme k dispozici; máme jen vzorek. Trénovací chyba je jako odhad optimisticky zkreslená, tím víc, čím víc se model datům přizpůsobí.

Další úlohy

- **Segmentace:** rozdělení vstupu na oblasti; v medicíně například vymezení nádoru na snímku.
- **Generování:** nová data podobná trénovacím — obrázky, text, zvuk, ... návrhy molekul.
- **Zpracování jazyka:** překlad, sumarizace, analýza sentimentu, dialogové systémy.
- **Rozpoznávání obrazu a zvuku:** obličeje, gesta, řeč.
- **Sekvenční rozhodování:** volba akcí agenta v prostředí.
- **Rekonstrukce dat:** odšumování, doplnění chybějících částí.

Pro každou z nich se vyvinula jiná architektura — konvoluční sítě pro obraz, rekurentní a později Transformer pro sekvence. Společné jim zůstává schéma *reprezentace – model – ztráta – metrika*.

Empirický odhad: dělení dat

- **trénovací množina** — upravují se na ní parametry,
- **validační množina** — volí se na ní architektura, síla regularizace, augmentace i okamžik zastavení,
- **testovací množina** — přijde na řadu až na konci.

Kdo se do testovací množiny dívá průběžně a podle výsledku mění postup, přeučí se i na ni, jen pomaleji než na trénovací data.

Při malém datasetu se používá k -násobná křížová validace; uvádí se průměr i rozptyl metrik mezi foldy. **Stratifikace** přibližně zachová zastoupení tříd.

Co se nesmí rozdělit napříč

Závislé příklady musí zůstat ve stejné části:

- snímky jednoho pacienta,
- věty jednoho dokumentu,
- navazující časové záznamy,
- fotografie z jedné výrobní série.

Jinak model vidí tutéž entitu při trénování i testu a výsledek je nadhodnocený.

Příklad 1 (Nepřímé přeučení na test)

Výzkumník vyzkouší dvacet modelů a vybere ten s nejvyšší testovací přesností. Tím z testu udělal validační množinu. I kdyby byly všechny modely stejně dobré, nejlepší z dvaceti zašuměných výsledků vypadá lépe, než jaká je skutečnost.

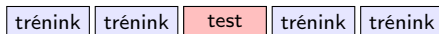
Křížová validace a bootstrap

k -násobná křížová validace

Data rozdělíme na k částí, model vždy trénujeme na $k - 1$ z nich a vyhodnotíme na zbývajících. Uvádíme průměr i rozptyl metrik.

Bootstrap

Z datasetu o n prvcích vytvoříme nový vzorek o n prvcích výběrem *s opakováním*. Nevybrané (*out-of-bag*) příklady slouží k odhadu chyby.



jeden fold z pěti; test se v dalších bězích posouvá

Samostatný testovací soubor, pokud jej máme, do tohoto cyklu nevstupuje.

Metrika musí odpovídat úloze

Příklad 2 (Accuracy u nevyvážených tříd)

Dataset obsahuje 990 bezvadných a 10 vadných výrobků. Model, který vždy predikuje „bez vady“, má accuracy $990/1000 = 99\%$, přestože nenajde jedinou vadu.

Užitečnější je matice záměn a z ní odvozené

$$\text{precision} = \frac{TP}{TP + FP}, \quad \text{recall} = \frac{TP}{TP + FN}.$$

Pomůže také vážená ztráta nebo posun rozhodovacího prahu — ale prahem se mění provozní bod, ne kvalita modelu.

Matice záměn a odvozené metriky

	predikce +	predikce -
skutečnost +	TP	FN
skutečnost -	FP	TN

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2PR}{P + R}$$

Příklad 3 (Dvě chyby nejsou totéž)

U detekce vady stojí přehlédnutí (*FN*) reklamaci, planý poplach (*FP*) jen kontrolu navíc. Volba prahu mezi precision a recall je proto provozní rozhodnutí, ne vlastnost modelu.

Nelinearita je nutná

Složením lineárních vrstev vznikne zase jen lineární zobrazení. Aktivační funkce proto musí být nelineární.

Věta 4 (Věta o univerzální aproximaci)

Nechť $\phi \in C(\mathbb{R}, \mathbb{R})$. Pak ϕ není polynom právě tehdy, když pro každé n, m , kompaktní $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m)$ a $\varepsilon > 0$ existují k , matice $\mathbf{A}$, $\mathbf{C}$ a vektor $\mathbf{b}$ tak, že

$$\sup_{\mathbf{x} \in K} \|\mathbf{f}(\mathbf{x}) - \mathbf{g}(\mathbf{x})\| < \varepsilon, \quad \mathbf{g}(\mathbf{x}) = \mathbf{C}(\phi \circ (\mathbf{A}\mathbf{x} + \mathbf{b})).$$

Jde o síť s *jednou* skrytou vrstvou a lineárním výstupem.

Jak se k větě došlo

- **1989:** Cybenko dokázal tvrzení pro sigmoidní aktivaci.
- **1991:** Hornik ukázal, že nejde o vlastnost sigmoidy, ale samotné vícevrstvé dopředné architektury; stačí spojitá, omezená a nekonzstantní aktivace.
- **1993:** Leshno a kol. dokázali, že nutnou a postačující podmínkou je nepolynomiálnost aktivační funkce.

Co věta neříká

Zaručuje, že hledaná síť *existuje*. Neříká, kolik bude mít jednotek (může jich být astronomicky mnoho), ani jak k jejím vahám dojít učením z dat. Obojí je v praxi obtížnější než samotná otázka existence.

G. Cybenko: Approximation by Superpositions of a Sigmoidal Function. *Math. of Control, Signals and Systems* 2(4) (1989). — K. Hornik: Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks* 4(2) (1991). — M. Leshno a kol.: *Neural Networks* 6(6) (1993).

Numerické a booleovské atributy

- **Booleovské** atributy kódujeme jako 0/1 (nebo $-1/+1$).
- **Numerické** atributy škálujeme do zvoleného intervalu nebo standardizujeme na nulový průměr a jednotkovou odchylku:

$$x' = \frac{x - \mu}{\sigma}.$$

- Liší-li se hodnoty řádově, pomůže *logaritmická* škála.
- Veličinu s přirozenými intervaly lze kvantizovat; ztrácíme tím ovšem přesnou hodnotu i uspořádání uvnitř intervalu.

Statistiky jen z trénovací části

Průměr a odchylku počítáme z trénovacích dat a stejná čísla použijeme všude jinde. Jinak informace z testu prosákne do trénování.

Kategorie: one-hot, ne pořadová čísla

Kategorickému atributu přiřadíme binární vektor s jedinou jedničkou:

červená $\mapsto (1, 0, 0)$, zelená $\mapsto (0, 1, 0)$, modrá $\mapsto (0, 0, 1)$.

Kódování jediným číslem $(1, 2, 3)$ by do dat vneslo umělé uspořádání a vzdálenosti, které v nich nejsou.

Při tisících hodnot vznikají dlouhé řidké vektory; tehdy se místo one-hot učí kratší hustý [embedding](#).

Příklad 5 (Tři atributy, tři různá kódování)

Barvu kódujeme one-hot. Věk standardizujeme jako $(x - \mu)/\sigma$. Identifikátor zákazníka jako číselný atribut nepoužijeme vůbec: síť by se mohla naučit konkrétní osoby a náhodné dělení dat by nadhodnotilo generalizaci.

Embedding místo dlouhého one-hot vektoru

Pro slovník o V hodnotách má one-hot vektor délku V a všechny dvojice kategorií jsou stejně vzdálené.

Embedding je naučená tabulka $E \in \mathbb{R}^{V \times d}$ s $d \ll V$; kategorii odpovídá řádek E_i . Vyhledání je jediné čtení z paměti a tabulka se učí spolu se zbytkem sítě.



Vzdálenosti v embeddingovém prostoru už nesou informaci o podobnosti kategorií. Podrobně u vektorových reprezentací slov v deváté přednášce.

Chybějící a odlehlé hodnoty

- **Chybějící hodnota** není nula. Doplnujeme ji explicitně (průměrem, mediánem, modelem) a přidáváme *indikátor chybějící hodnoty* jako další vstup.
- **Odehlá hodnota** může být chyba měření i podstatný jev. Ořezání (*winsorizace*) nebo logaritmická škála pomáhají, ale mění data — rozhodnutí patří do dokumentace.
- Statistiky pro doplnění i ořezání se počítají *jen* z trénovací části.

Tichá chyba

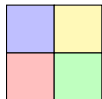
Doplnění nulou u standardizovaných dat znamená „průměrná hodnota“, zatímco u nestandardizovaných „nula“. Táž operace tedy má v různých místech pipeline různý význam.

Obraz: proč jej nezploštit

Obraz $X \times Y$ pixelů zapisujeme jako tenzor $X \times Y \times 3$.

Zploštěním na vektor délky $3XY$ bychom hodnoty zachovali, ale architektura by už neměla přímo k dispozici informaci o sousedství pixelů ani o tom, že tři kanály patří k témuž pixelu.

$2 \times 2 \times 3$



vektor délky 12

Prostorovou strukturu zachovávají konvoluční sítě pomocí lokálních spojení a sdílených vah — k nim se dostaneme v sedmé přednášce.

Od kvadratické chyby k věrohodnosti

Kvadratická chyba udržuje výpočty jednoduché, ale u klasifikace je přirozenější číst výstup jako pravděpodobnost a maximalizovat věrohodnost dat.

Pro jeden příklad je ztrátou záporný logaritmus pravděpodobnosti správné třídy

$$\mathcal{L} = -\log P_{\mathbf{w}}(\mathbf{t}_j | \mathbf{x}_j),$$

pro celou množinu pak

$$-\sum_{j=1}^N \log P_{\mathbf{w}}(\mathbf{t}_j | \mathbf{x}_j).$$

Čím blíže je přiřazená pravděpodobnost jedničce, tím blíže je ztráta nule; pravděpodobnost blízka nule vede k velké ztrátě.

Softmax

Pro K tříd potřebujeme K nezáporných výstupů se součtem 1:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}.$$

Vztah k sigmoidě

Pro $K = 2$ dostaneme po vydělení e^{z_1}

$$p_1 = \frac{1}{1 + e^{z_2 - z_1}} = \sigma(z_1 - z_2),$$

softmax je tedy zobecnění sigmoidy na více tříd.

Ve dvojici s kategoričnou křížovou entropií tvoří softmax standardní zakončení sítě pro vícetřídní klasifikaci.

Numerická stabilita

Počítá se až po odečtení maxima,

$$p_i = \frac{\exp(z_i - m)}{\sum_j \exp(z_j - m)}, \quad m = \max_j z_j,$$

jinak e^z snadno přeteče.

Softmax jako vyhlazený argmax

Funkce max je pro trénování nepoužitelná: není hladká a její gradient je téměř všude nulový. Hladkou náhradou je **log-sum-exp**

$$\text{LSE}(\mathbf{x}) = \log \sum_{j=1}^d e^{x_j}, \quad \max_j x_j \leq \text{LSE}(\mathbf{x}) \leq \max_j x_j + \log d.$$

Se softmaxem souvisí derivováním:

$$\frac{\partial}{\partial x_i} \text{LSE}(\mathbf{x}) = \frac{e^{x_i}}{\sum_j e^{x_j}} = \text{softmax}(\mathbf{x})_i.$$

Softmax je tedy vyhlazenou verzí funkce *argmax*: jeho výstupem není největší hodnota, ale měkký ukazatel na to, která složka je největší.

Křížová entropie

Pro rozdělení P (cíl) a Q (model) je

$$H(P, Q) = H(P) + D_{\text{KL}}(P \| Q),$$

takže pro pevné P je minimalizace křížové entropie totéž co minimalizace KL divergence.

Příklad 6 (Výpočet pro tři příklady)

Model klasifikuje sentiment do tří tříd a správné třídě přiřadil pravděpodobnosti 0,7, 0,8 a 0,5. Ztráta je

$$\mathcal{L} = -[\log 0,7 + \log 0,8 + \log 0,5] \doteq 1,27.$$

Je-li cílové rozdělení one-hot, zbude jediný člen — záporný logaritmus pravděpodobnosti správné třídy. Křížová entropie je tedy zobecněním záporné logaritmické věrohodnosti na obecné cílové rozdělení.

Proč se softmax a křížová entropie párují

Pro cíl $\mathbf{t}$ v one-hot kódování a logity $\mathbf{z}$ je ztráta $\mathcal{L} = -\log p_c$, kde c je správná třída. Derivace podle logitu vyjde pozoruhodně jednoduše:

$$\frac{\partial \mathcal{L}}{\partial z_i} = p_i - t_i.$$

- Gradient je přímo *rozdíl predikce a cíle*; žádné další faktory z aktivace nezůstávají.
- Nehrozí tedy útlum, který by vznikl, kdybychom sigmoidu nebo softmax kombinovali s kvadratickou chybou.
- Odtud také plyne, že knihovny počítají softmax a ztrátu společně z logitů: je to numericky stabilnější i levnější.

Pro binární případ platí totéž se sigmoidou: $\partial \mathcal{L} / \partial z = \sigma(z) - t$.

Přehled: úloha, výstup, ztráta

úloha	výstupní vrstva	ztráta
binární klasifikace	1 jednotka, sigmoida	binární křížová entropie
vícetřídní klasifikace	K jednotek, softmax	kategorická křížová entropie
více nezávislých značek	K jednotek, sigmoida	součet binárních CE
regrese	lineární výstup	MSE (nebo MAE)
regrese s nejistotou	μ a σ^2	Gaussovská NLL

Častá chyba

Softmax a křížová entropie se v knihovnách obvykle počítají *společně* z logitů. Aplikovat softmax ručně a pak ještě jednou uvnitř ztráty znamená počítat jej dvakrát.

Metriky regrese

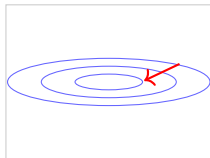
$$\text{MSE} = \frac{1}{N} \sum_i (y_i - f(x_i))^2, \quad \text{MAE} = \frac{1}{N} \sum_i |y_i - f(x_i)|,$$

$$R^2 = 1 - \frac{\sum_i (y_i - f(x_i))^2}{\sum_i (y_i - \bar{y})^2}.$$

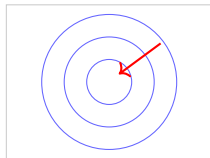
- MSE trestá velké odchylky kvadraticky, a je proto citlivá na odlehlé hodnoty.
- MAE je robustnější, ale její gradient nezávisí na velikosti chyby.
- R^2 vztahuje chybu k rozptylu dat; sama o sobě neříká, zda je model pro danou aplikaci dost přesný.

Proč na měřítku vstupů záleží

Mají-li dva atributy velmi různý rozsah, je vrstevnicová mapa ztráty protáhlá: gradient míří napříč údolím, ne k minimu.



bez škálování



po standardizaci

Standardizace vstupů proto není kosmetika: mění tvar optimalizační úlohy. Stejnou roli hrají uvnitř sítě normalizační vrstvy, o nichž bude řeč v šesté přednášce.

Mizející a explodující gradienty

Gradient vzdálené vrstvy obsahuje součin mnoha derivací.

Mizení

Derivace sigmoidy a tanh mají absolutní hodnotu menší než 1, takže součin s hloubkou exponenciálně klesá. Vrstvy u vstupu se přestanou učit.

Exploze

Součin naopak rychle roste, typicky u rekurentních sítí, kde se opakuje tatáž matice vah. Trénování osciluje nebo diverguje.

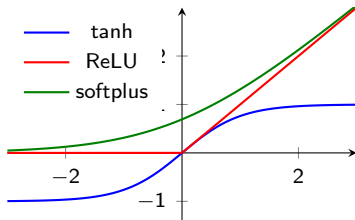
Explozi lze omezit ořezáváním gradientu podle normy. Na mizení má vliv volba aktivační funkce — proto se od sigmoidy ve skrytých vrstvách ustoupilo.

Historicky: 1985–2010 sigmoida a tanh; od roku 2010 převážně ReLU a její varianty.

Od tanh k ReLU

$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ je souměrný kolem nuly, takže aktivace nemají systematicky kladný průměr.

$\text{ReLU}(x) = \max(0, x)$ nemá horní saturaci, takže pro kladné vstupy gradient neklesá. V nule není hladká; derivace se tam konvencí volí 0.



V. Nair, G. E. Hinton: Rectified Linear Units Improve Restricted Boltzmann Machines. ICML (2010). — C. Dugas a kol.: Incorporating Second-Order Functional Knowledge for Better Option Pricing. NIPS (2001).

Varianty ReLU

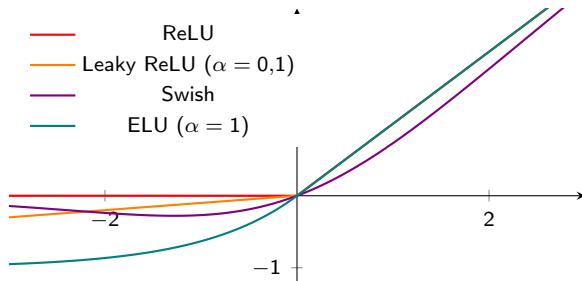
funkce	předpis	poznámka
ReLU	$\max(0, x)$	levná; jednotka může odumřít
Leaky ReLU	$\max(\alpha x, x), \alpha \approx 0,01$	záporná větev má derivaci α
softplus	$\ln(1 + e^x)$	hladká, dražší; kladný výstup
ELU	$x \text{ resp. } \alpha(e^x - 1)$	záporná větev stahuje průměr k nule
Swish	$x \sigma(x)$	hladká, z automatizovaného hledání

Odumřelá jednotka

Má-li ReLU jednotka trvale záporné preaktivace, je její gradient nulový a už se nikdy neaktivuje. Leaky ReLU a ELU tomu předcházejí nenulovou zápornou větví.

A. L. Maas a kol.: Rectifier Nonlinearities Improve Neural Network Acoustic Models. ICML (2013). — D.-A. Clevert, T. Unterthiner, S. Hochreiter: ELU. ICLR (2016). — P. Ramachandran, B. Zoph, Q. V. Le: Searching for Activation Functions. arXiv:1710.05941 (2017).

Varianty graficky



Rozdíly jsou největší v okolí nuly a v záporné větvi; pro velká kladná x se všechny blíží identitě.

Sigmoida na výstupu versus ve skryté vrstvě

Na výstupu

Má jasný význam: převádí logit na pravděpodobnost. Saturace nevadí, protože ztráta (křížová entropie) ji kompenzuje — gradient je $\sigma(z) - t$.

Ve skryté vrstvě

Saturace se sčítá přes vrstvy a gradient mizí. Navíc jsou aktivace vždy kladné, takže mají nenulový průměr, což zpomaluje učení.

Táž funkce tedy může být na jednom místě architektury vhodná a na jiném ne. Volba aktivace není otázka vkusu, ale toho, co má vrstva dělat.

Počet a velikost skrytých vrstev

Hloubka dovoluje opakovaně skládat jednodušší transformace: časné vrstvy reagují na hrany, další jejich výstupy kombinují do složitějších tvarů.

Není však obecně pravda, že při stejném počtu parametrů je hlubší síť vždy lepší; záleží na úloze, architektuře a optimalizaci.

Praktický důsledek

Teorie, která by pro daný problém dopředu určila vhodnou architekturu, neexistuje. Návrh sítě je z velké části empirický: vyplatí se umět rychle vyzkoušet několik variant a poctivě je porovnat.

Co znamená „poctivě“, rozebereme v přednášce o generalizaci a regularizaci.

Cvičení k přednášce 3

1. ★ Pro úlohu predikce ceny zvolte výstupní vrstvu, ztrátovou funkci a dvě evaluační metriky.
2. ★ Totéž proveďte pro binární detekci podvodu a vícetřídní klasifikaci obrazu.
3. ★ Spočítejte sigmoidní a softmaxový výstup pro logity $(2, 1, 0)$.
4. ★★ Zakódujte atributy barva, věk a PSČ a zdůvodněte každou reprezentaci.
5. ★★ Uveďte tři příklady úniku informace při konstrukci vstupu.
6. ★★ Ukažte, že pro dvě třídy dává softmax totéž co sigmoida rozdílu logitů.
7. ★★ Spočítejte křížovou entropii pro cíl $(0, 1, 0)$ a predikci $(0,2, 0,5, 0,3)$.
8. ★★★ Proč se softmax počítá až po odečtení maxima logitů? Uveďte příklad, kdy bez toho výpočet selže.
9. ★★★ Navrhněte metriku pro úlohu, kde je cena falešně negativní stokrát vyšší než cena falešně pozitivní.