Questions

Q1.1 Explain how reinforcement learning differs from supervised and unsupervised learning in terms of the type of input the learning algorithms use to improve model performance. [5]

Odpověď (česky):

Posilované učení (Reinforcement Learning):

- Algoritmus se učí na *zpětné vazbě* ve formě odměn (reward) či penalizací.
- Cílem je maximalizovat kumulovanou odměnu při interakci s prostředím.
- Typické použití: hry, robotika, řízení systémů. Agent provádí akce v prostředí a učí se na základě dosažených odměn.

Učení s učitelem (Supervised Learning):

- Dostáváme trénovací vzorky *vstupů* spolu se správnými *výstupy*.
- Model se učí mapovat vstup na výstup ($x \to t$).
- Typické úlohy: klasifikace, regrese.

Učení bez učitele (Unsupervised Learning):

- Máme pouze *neoznačená* vstupní data.
- Cílem je objevit skrytou strukturu (shlukování, detekce anomálií, redukce rozměru) bez explicitních labelů.

Hlavní rozdíl: V posilovaném učení se model adaptuje na základě dlouhodobých odměn (neexistují explicitní správné výstupy pro každý krok), zatímco učení s učitelem pracuje s předem danými výstupy a učení bez učitele hledá struktury v datech bez jakýchkoliv výstupních značek.

Q1.2 Explain why we need separate training and test data. What is generalization, and how does the concept relate to underfitting and overfitting? [10]

Odpověď (česky):

Proč potřebujeme trénovací a testovací data:

- *Vyhodnocení výkonu na neviděných datech:* Testovací sada slouží jako reálný indikátor toho, jak dobře model generalizuje.
- *Prevence overfittingu:* Používáme-li stejná data pro trénování i hodnocení, hrozí memorování dat bez obecného přenosu na nová data.
- *Lepší ladění hyperparametrů:* Pro volbu a úpravu modelu (např. výběr regularizace) potřebujeme nezávislou sadu.

Co je generalizace:

- Schopnost modelu *přenést získané znalosti* na nová, dosud neviděná data.
- Základem dobré generalizace je učení skutečných struktur místo pouhého zapamatování trénovacích příkladů.

Underfitting a overfitting:

- **Underfitting:** Model je příliš jednoduchý, nedokáže zachytit skutečné vzory v datech. Výsledkem je slabý výkon jak na tréninkové, tak na testovací sadě.
- **Overfitting:** Model je příliš komplexní, naučí se i šum trénovací sady a zobecní špatně na testovacích datech.

Q1.3 Define the prediction function of a linear regression model and write down L^2-regularized mean squared error loss. [10]

Odpověď (česky):

Predikční funkce lineární regrese:

$$f(x; w, b) = x^T w + b,$$

kde $x \in \mathbb{R}^D$ je vstupní vektor, w je váhový vektor a b je skalární bias.

L^2-regulovaná střední kvadratická ztráta (Ridge Regression):

$$L(w, b) = \frac{1}{2} \sum_{i=1}^{N} \Big(f(x_i; w, b) - t_i \Big)^2 + \frac{\lambda}{2} \|w\|^2,$$

kde t_i je cílová hodnota pro i-tý vzorek, λ je regularizační parametr a $\|w\|^2$ označuje součet čtverců složek vektoru w.

Q1.4 Starting from the unregularized sum of squares error of a linear regression model, show how the explicit solution can be obtained, assuming $X^T X$ is invertible. [10]

Odpověď (česky):

Nechť funkce chyby pro lineární regresi bez regularizace zní

$$E(w) = \frac{1}{2} \sum_{i=1}^{N} \big(x_i^T w - t_i \big)^2.$$

Minimalizaci hledáme položením gradientu k nule:

$$\nabla_w E(w) = X^T (Xw - t) = 0,$$

kde X je matice vstupů $(N \times D)$ a t je vektor cílových hodnot $(N \times 1)$.

Výsledná podmínka je

$$X^T X \, w = X^T t.$$

Pokud je $(X^T X)$ *invertibilní*, získáme explicitní řešení:

$$w = (X^T X)^{-1} X^T t.$$

Part II

Lecture 2

Q2.1 Describe standard gradient descent and compare it to stochastic (i.e., online) gradient descent and minibatch stochastic gradient descent. [10]

Odpověď (česky):

Standard Gradient Descent (Batch GD):

- Také nazývaný *batch gradient descent*, spočítá gradient chybové funkce *pro celou* trénovací množinu.
- Aktualizace vah probíhá dle vzorce

$$w \leftarrow w - \alpha \nabla_w E(w),$$

  kde α je učicí rychlost (learning rate) a $E(w)$ je chybová funkce.
- Výhodou je stabilní odhad gradientu, nevýhodou mohou být vysoké nároky na výpočet pro velké množiny dat.

Stochastic Gradient Descent (SGD, tzv. online učení):

- Aktualizace vah se provádí *po každém* trénovacím vzorku:

$$\nabla_w E(w) \approx \nabla_w L\big(y(x_i; w), t_i\big),$$

  kde x_i je vstup a t_i cíl pro jediný vzorek.
- Je *rychlejší* na jednu aktualizaci (z důvodu malých datových dávek), ale má vyšší šum v odhadu gradientu.
- Pro velmi velké datasety často konverguje rychleji než batch GD.

Minibatch SGD:

- *Kompromis* mezi batch GD a čistě stochastickým přístupem.
- Aktualizace vah se provádí pro malou dávku (batch) B trénovacích vzorků:

$$\nabla_w E(w) \approx \frac{1}{|B|} \sum_{i \in B} \nabla_w L\big(y(x_i; w), t_i\big).$$

- Dosahuje kompromisu mezi stabilitou odhadu gradientu a rychlostí konvergence.

Q2.2 Write an L^2-regularized minibatch SGD algorithm for training a linear regression model, including the explicit formulas of the loss function and its gradient. [10]

Odpověď (česky):

L^2-regulovaná ztrátová funkce pro lineární regresi:

$$E(w) \;=\; \mathbb{E}_{(x,t)\sim\,\hat{p}_{\text{data}}}\big[\frac{1}{2}(x^T w - t)^2\big] \;+\; \frac{\lambda}{2}\,\|w\|^2,$$

kde w je vektor vah, x vstup, t cílová hodnota a λ je regulační parametr.

Gradient ztrátové funkce vzhledem k w:

$$\nabla_w E(w) \;\approx\; \frac{1}{|B|}\sum_{i\in B}\big((x_i^T w - t_i)\,x_i\big) \;+\; \lambda\,w,$$

kde B je *minibatch* (podmnožina trénovacích vzorků).

Algorithm 1 Pseudokód minibatch SGD algoritmu

Require: Dataset $\{X \in \mathbb{R}^{N\times D}, t \in \mathbb{R}^N\}$, learning rate $\alpha \in \mathbb{R}^+$, regulační parametr $\lambda \in \mathbb{R}$

Ensure: $w \in \mathbb{R}^D$ minimalizující regularizovanou MSE lineární regrese

1: Inicializuj w náhodně
2: **repeat**
3: Zvol minibatch B (podmnožina trénovacích příkladů) s indexy B
4: Spočti gradient $\mathbf{g}$ podle $\nabla_w E(w)$ využitím minibatche B
5: Aktualizuj váhy: $w \leftarrow w - \alpha \cdot \mathbf{g}$
6: **until** konvergence nebo dosažení maximálního počtu iterací

Q2.3 Does the SGD algorithm for linear regression always find the best solution on the training data? If yes, explain under what conditions it happens, if not explain why it is not guaranteed to converge. [20]

Odpověď (česky):

Stochastický gradientní sestup (SGD) pro lineární regresi *nemusí* vždy zaručit nalezení nejlepšího (globálního) řešení na trénovacích datech. Konverguje k globálnímu optimu, pokud jsou splněny následující podmínky:

- **Konvexní a spojitá ztrátová funkce:** Pokud má problém jediné minimum, lze zaručit konvergenci k tomuto minimu.
- **Učicí rychlost α_i splňuje Robbins-Monro podmínky:**

$$\alpha_i > 0, \quad \sum_{i=1}^{\infty}\alpha_i = \infty, \quad \sum_{i=1}^{\infty}\alpha_i^2 < \infty,$$

přičemž poslední z těchto podmínek zajišťuje, že $\alpha_i \to 0$ pro $i \to \infty$.

Pokud jsou tyto předpoklady splněny, *SGD* se může dostat k jedinému (globálnímu) minimu konvexních problémů. Nicméně u **nekonvexních** ztrátových funkcí může algoritmus skončit v lokálním minimu. Navíc **stochastičnost** gradientu (šum) může konvergenci zpomalit či kolísat, což někdy vede k nehledání nejlepšího řešení. Přesto se v praxi, zejména pro velmi velké datasety, SGD často používá a poskytuje dobré výsledky, i když *záruka* nalezení absolutně nejlepšího řešení nemusí platit.

Q2.4 After training a model with SGD, you ended up with a low training error and a high test error. Using the learning curves, explain what might have happened and what steps you might take to prevent this from happening. [10]

Odpověď (česky):

Z křivek učení lze usoudit, že trénovací ztráta se postupně snižuje, zatímco testovací ztráta nejprve klesá, ale po čase začne stoupat. To naznačuje, že model *přeučil* (overfit) na trénovací data. Přeučení nastává, když se model naučí i šum a detaily specifické pro trénovací sadu, které se neobecně promítnou na neviděné (testovací) vzorky. Výsledkem je, že model má velmi dobrý výkon na trénovací sadě, ale špatný výkon na testovací sadě.

Jak overfittingu předcházet:

1. **Regulace vah:** Použít např. L_1 (LASSO) nebo L_2 (Ridge) regulaci k penalizaci příliš velkých vah.
2. **Early stopping:** Na základě výkonu na validační sadě zastavit učení dříve, než dojde k výraznému přeučení.
3. **Rozšíření trénovací sady:** Pokud je to možné, zvýšit počet trénovacích vzorků pro lepší zobecnění.
4. **Zjednodušení modelu:** Omezit složitost (např. použít méně vrstev v neurální síti, snížit počet parametrů, vybrat menší počet příznaků).

Tyto metody pomáhají modelu *obecněji* reagovat na dosud neviděná data a zlepšit tak výkon na testovací sadě. Dalším důvodem vysoké testovací chyby může být i *selhání konvergence* (model se plně „nedoučil“). V takovém případě lze zkusit zvýšit počet trénovacích iterací nebo upravit učicí rychlost, aby se zlepšila konvergence.

Q2.5 You were provided with a fixed training set and a fixed test set and you are supposed to report model performance on that test set. You need to decide what hyperparameters to use. How will you proceed and why? [5]

Odpověď (česky):

Pro nalezení optimálních hyperparametrů, když máme pevně danou trénovací a testovací sadu, se běžně postupuje takto:

1. **Rozdělení trénovací sady:** Část trénovacích dat vyhradíme jako *validační* sadu (training–validation split).

2. **Hledání hyperparametrů:** Na menší trénovací sadě zkoušíme různé kombinace hyperparametrů (např. grid search, random search, Bayesian optimalizace atd.).

3. **Vyhodnocení na validační sadě:** Každý model ověříme na validační sadě.

4. **Výběr nejlepší konfigurace:** Zvolíme hyperparametry, které si na validační sadě vedly nejlépe.

5. **Finální model:** Trénujeme nový model s vybranými hyperparametry na *celé* trénovací sadě.

6. **Test:** Vyhodnotíme výkon na *pevně dané* testovací sadě.

Tento proces je důležitý, protože nám umožní *nepřetrénovat* (overfit) na samotnou trénovací sadu během ladění hyperparametrů. Validační sada tak slouží jako zástupce pro reálný test, na kterém testujeme, jak dobře nastavení modelu funguje na neviděných datech.

Q2.6 What method can be used for normalizing feature values? Explain why it is useful. [5]

Odpověď (česky):

Min-Max Normalizace:

$$x'_{i,j} = \frac{x_{i,j} - \min_k x_{k,j}}{\max_k x_{k,j} - \min_k x_{k,j}}$$

- Škáluje featury do pevného intervalu, např. $[0, 1]$.
- Hodí se v případech, kdy chceme zachovat relativní odstupy, ale omezit hodnoty do určitého rozsahu.

Z-score Standardizace:

$$x'_{i,j} = \frac{x_{i,j} - \overline{x}_j}{\sigma_j},$$

- Převede featury na takové, které mají střední hodnotu (mean) 0 a směrodatnou odchylku 1.
- Často užitečné v optimalizačních algoritmech, které předpokládají příznaky na srovnatelné škále.

Proč normalizovat:

- Zrychluje a stabilizuje trénovací proces (zejména gradientní metody), protože příliš vysoké rozdíly v měřítku příznaků mohou vést k nevyváženému učení.
- Redukuje šum a duplicity: Další pokročilé metody (např. PCA) pak mohou ještě zmenšit rozměrnost a odstranit redundantní informace.

Q2.7 Explain possible intuitions behind L^2 regularization. [5]

Odpověď (česky):

Intuice a význam L^2 regularizace (Ridge):

- **Penalizace velkých vah:** Regulační člen $\lambda \|w\|^2$ upřednostňuje „menší“ hodnoty prvků váhového vektoru w. To snižuje riziko, že se model naučí extrémní hodnoty vah, které často vedou k přeučení.
- **Vyhlazení a stabilizace modelu:** Přidání L^2 penalizace znamená, že model je méně náchylný k výkyvům (šumu) v trénovacích datech. Výsledkem je obecnější chování (lepší generalizace).
- **Geometrická interpretace:** L^2 regularizace „přitahuje“ řešení k počátku v prostoru vah. Model se tak udržuje v menší oblasti vah, což pomáhá snížit komplexitu a zabránit overfittingu.

Q2.8 Explain the difference between hyperparameters and parameters. [5]

Odpověď (česky):

- **Parametry (parameters):**
  - Uvnitř modelu, učí se *přímo* z dat (např. váhy neuronové sítě, koeficienty lineární regrese).
  - Bývají průběžně upravovány během učení (např. gradientním sestupem) a jsou klíčem k mapování vstupu na výstup.
- **Hyperparametry (hyperparameters):**
  - *Nastavujeme je před trénováním* a *neučí* se z dat přímo (např. učicí rychlost α, počet skrytých vrstev v síti, regulační koeficient λ).
  - Ovlivňují, jak se parametry modelu trénují, případně ovlivňují kapacitu modelu (jeho složitost) či rychlost konvergence.
  - Typicky se ladí pomocí *validační* sady, křížové validace či jiných postupů (grid search, random search, Bayesian optimalizace apod.).

Hlavní rozdíl: Parametry jsou skutečné *učící se* veličiny modelu, zatímco hyperparametry jsou *nastavení*, která určují, jak (a v jaké velikosti) se učení provádí. Parametry tedy model *drží v paměti* pro práci s daty, zatímco hyperparametry jsou spíše „ovladače“ (konfigurace), jež musíme zvolit ještě před spuštěním procesu učení.

Part III

Q3.1 Define binary classification, write down the perceptron algorithm and show how a prediction is made for a given example. [10]

Odpověď (česky):

Binární klasifikace:

- Úloha rozdělit vstupní vzorky do *dvou* kategorií (tříd), označovaných např. jako $\{0, 1\}$ nebo $\{-1, +1\}$.
- Typické příklady: Rozhodnout, zda je e-mail spam/ne-spam, nebo zda pacient má/nemá určitou nemoc apod.

Perceptron algoritmus:

- Perceptron je jednoduchý lineární klasifikátor.
- Má váhový vektor w a bias (případně můžeme bias zahrnout do váhového vektoru).
- Během trénování, pro každý trénovací příklad (x_i, t_i), kde $t_i \in \{-1, +1\}$, provádíme:

$$\text{if } t_i \left(x_i^T w\right) \leq 0 \quad \text{then} \quad w \leftarrow w + t_i\, x_i \quad \text{(aktualizace vah)}.$$

Predikce pro nový příklad:

$$\hat{y} \;=\; \operatorname{sign}\bigl(x^T w + b\bigr),$$

kde $\operatorname{sign}(z)$ vrátí $+1$ pro kladný vstup a -1 pro záporný. Pokud tedy výsledek skalárního součinu (plus bias) je kladný, příklad je přiřazen k jedné třídě; pokud je záporný, přiřadíme ho k druhé třídě.

Q3.2 Define entropy, cross-entropy, Kullback-Leibler divergence, and prove the Gibbs inequality. [20]

Odpověď (česky):

Entropie $H(P)$:

$$H(P) = -\sum_x P(x)\, \log P(x).$$

- Udává „míru překvapení“ či nejistoty, související s rozdělením P.
- Vyšší entropie znamená větší neurčitost ohledně výsledků náhodné proměnné.

Křížová entropie $H(P, Q)$:

$$H(P, Q) = -\sum_x P(x)\, \log Q(x).$$

- Vyjadřuje očekávaný počet bitů (pokud log je o základu 2) potřebných k rozeznání určité události, používáme-li rozdělení Q namísto skutečného P.
- Čím je křížová entropie menší, tím lépe Q aproximuje P.

Kullback-Leiblerova divergence $D_{\mathrm{KL}}(P\|Q)$:

$$D_{\mathrm{KL}}(P\|Q) = H(P, Q) - H(P) = \sum_x P(x)\, \log \frac{P(x)}{Q(x)}.$$

- Měří, o kolik se jedno rozdělení P liší od druhého (Q).

- Není symetrická ($D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$) a platí $D_{\mathrm{KL}}(P\|Q) \geq 0$.

Gibbsova nerovnost (důkaz):

$$H(P,Q) \;\geq\; H(P), \quad \text{tedy} \quad H(P) = H(P,Q) \iff P = Q.$$

Důkaz: Zvažme rozdíl

$$H(P) - H(P,Q) = \sum_x P(x)\, \log \frac{Q(x)}{P(x)}.$$

Použijeme nerovnost $\log z \leq (z-1)$ (s rovností jen pro $z=1$):

$$\sum_x P(x)\, \log \frac{Q(x)}{P(x)} \;\leq\; \sum_x P(x) \left(\frac{Q(x)}{P(x)} - 1\right) \;=\; \sum_x Q(x) - \sum_x P(x) \;=\; 0.$$

Z toho plyne

$$\sum_x P(x)\, \log \frac{P(x)}{Q(x)} \;\geq\; 0,$$

což je totéž co $D_{\mathrm{KL}}(P\|Q) \geq 0$. Rovnost nastává výlučně, když $\frac{Q(x)}{P(x)} = 1$ pro každé x, tj. $P(x) = Q(x)$ pro všechna x. Tím je Gibbsova nerovnost dokázána.

Q3.3 Explain the notion of likelihood in maximum likelihood estimation. [5]

Pravděpodobnost (likelihood) v kontextu MLE:

Mějme trénovací data $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ získaná z datagenerujícího rozdělení p_{data}. Předpokládejme, že máme rodinu modelů $p_{\mathrm{model}}(\mathbf{x}; w)$. Pak *likelihood* pro parametry w je:

$$L(w) \;=\; p_{\mathrm{model}}\big(\mathbf{X}; w\big) \;=\; \prod_{i=1}^{N} p_{\mathrm{model}}(x_i; w).$$

- **Maximum Likelihood Estimation (MLE)** hledá

  $$w_{\mathrm{MLE}} \;=\; \arg\max_w \; p_{\mathrm{model}}\big(\mathbf{X}; w\big),$$

  což je ekvivalentní $\arg\max_w \prod_{i=1}^{N} p_{\mathrm{model}}(x_i; w)$.

- Likelihood (na rozdíl od pravděpodobnosti rozdělení nad w) je *skórovací funkce* měřící, jak „dobře“ parametry w vysvětlují pozorovaná data $\mathbf{X}$.

- Maximalizací $L(w)$ ($\neq$ minimalizací záporného log-likelihoodu) dosáhneme parametru, pro který jsou data $\mathbf{X}$ *nejpravděpodobnější* v rámci daného modelu p_{model}.

Q3.4 Describe maximum likelihood estimation, as minimizing NLL, cross-entropy, and KL divergence. [20]

Odpověď (česky):

Mějme trénovací data $X = \{x_1, x_2, \ldots, x_N\}$ nezávisle vybraná z datagenerujícího rozdělení p_{data}. Empirickou distribuci těchto dat aproximujeme pomocí $\hat{p}_{\text{data}}(x)$, kde

$$\hat{p}_{\text{data}}(x) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[x_i = x].$$

Náš model je parametrizován jako $p_{\text{model}}(x; w)$, kde w jsou parametry modelu, které se snažíme optimalizovat.

Maximum Likelihood Estimation (MLE) se snaží nalézt parametry w, které maximalizují pravděpodobnost trénovacích dat pod modelem:

$$w_{\text{MLE}} = \arg\max_w \prod_{i=1}^{N} p_{\text{model}}(x_i; w).$$

Pro snazší optimalizaci používáme logaritmickou transformaci:

$$w_{\text{MLE}} = \arg\min_w \Big(-\sum_{i=1}^{N} \log p_{\text{model}}(x_i; w) \Big).$$

Tento výraz minimalizuje **negativní log-likelihood (NLL)**, což odpovídá měření, jak dobře model vysvětluje data.

NLL a křížová entropie: Minimalizace NLL je ekvivalentní minimalizaci **křížové entropie** mezi empirickou distribucí $\hat{p}_{\text{data}}$ a modelovou distribucí p_{model}:

$$H(\hat{p}_{\text{data}}, p_{\text{model}}) = -\mathbb{E}_{x \sim \hat{p}_{\text{data}}}[\log p_{\text{model}}(x; w)].$$

NLL a KL-divergence: Minimalizace NLL je (až na konstantu) také minimalizací **KL-divergence** mezi $\hat{p}_{\text{data}}$ a p_{model}:

$$D_{\text{KL}}(\hat{p}_{\text{data}} \| p_{\text{model}}) = H(\hat{p}_{\text{data}}, p_{\text{model}}) - H(\hat{p}_{\text{data}}).$$

Zde $H(\hat{p}_{\text{data}})$ je konstantní entropie empirické distribuce, takže optimalizace w minimalizuje pouze křížovou entropii.

Podmíněný případ: Pro modelování $p_{\text{model}}(t \mid x; w)$ (např. v logistické regresi) je cílem maximalizovat:

$$\prod_{i=1}^{N} p_{\text{model}}(t_i \mid x_i; w),$$

což odpovídá minimalizaci podmíněné NLL:

$$-\sum_{i=1}^{N} \log p_{\text{model}}(t_i \mid x_i; w),$$

a opět i křížové entropie mezi $\hat{p}_{\text{data}}(t \mid x)$ a $p_{\text{model}}(t \mid x; w)$ a KL-divergence.

Stručné shrnutí: Maximum Likelihood Estimation optimalizuje parametry w tak, aby modelová distribuce $p_{\text{model}}(x; w)$ co nejlépe aproximovala empirickou distribuci $\hat{p}_{\text{data}}(x)$. Tohoto cíle dosahuje minimalizací negativního log-likelihoodu (NLL), což je ekvivalentní minimalizaci křížové entropie a (až na konstantu) KL-divergence.

Q3.5 Considering binary logistic regression model, write down its parameters (including their size) and explain how prediction is performed (including the formula for the sigmoid function). Describe how we can interpret the outputs of the linear part of the model as logits. [10]

Odpověď (česky):

Parametry modelu

- $\mathbf{w} \in \mathbb{R}^D$: váhový vektor (jeden váhový koeficient na každý z D vstupních příznaků).
- $b \in \mathbb{R}$: skalární bias (posun).

Model: pravděpodobnost třídy Logistická regrese pro binární klasifikaci modeluje pravděpodobnost, že příklad $\mathbf{x}$ náleží do třídy C_1:

$$p(C_1 \mid \mathbf{x}; \mathbf{w}) \;=\; \sigma\big(\mathbf{x}^T\mathbf{w} + b\big),$$

kde σ je *sigmoid* (logistická funkce):

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Logická doplňková pravděpodobnost je

$$p(C_0 \mid \mathbf{x}; \mathbf{w}) \;=\; 1 - p(C_1 \mid \mathbf{x}; \mathbf{w}).$$

Interpretace logits

- $\text{logit}(\mathbf{x}) \;=\; \mathbf{x}^T\mathbf{w} + b.$
- Lze ji číst jako $\log\big(\frac{p}{1-p}\big)$, tedy logaritmické šance třídy C_1 proti C_0.
- Sigmoid převádí reálné číslo $\text{logit}(\mathbf{x})$ na interval $[0, 1]$ a dává tak interpretaci jako „pravděpodobnost“ třídy C_1.

Predikce

$$\hat{y}(\mathbf{x}) \;=\; \begin{cases} 1, & \text{pokud } \sigma\big(\mathbf{x}^T\mathbf{w} + b\big) \geq 0.5, \\ 0, & \text{jinak.} \end{cases}$$

Tedy model přiřadí třídu C_1 těm vstupům, pro které je pravděpodobnost $p(C_1|\mathbf{x})$ (tj. výstup sigmoidu) alespoň 0.5.

Chybová (ztrátová) funkce Abychom odhadli $\mathbf{w}$ a b, minimalizujeme *negativní log-likelihood* (NLL), tj. průměrnou logistickou ztrátu:

$$E(\mathbf{w}) \;=\; -\frac{1}{N}\sum_{i=1}^{N} \log\, p\big(C_{t_i}\;\mathbf{x}_i; \mathbf{w}\big) \;=\; -\frac{1}{N}\sum_{i=1}^{N} \Big[t_i\, \log\, \sigma(z_i) \;+\; (1 - t_i)\, \log\big(1 - \sigma(z_i)\big)\Big],$$

kde $z_i \;=\; \mathbf{x}_i^T\mathbf{w} + b$ a $t_i \in \{0, 1\}$. Minimalizace této chyby se obvykle provádí pomocí *stochastického gradientního sestupu* (SGD) nebo jeho variant.

Q3.6 Write down an L^2-regularized minibatch SGD algorithm for training a binary logistic regression model, including the explicit formulas (i.e., formulas you would need to code it in numpy) of the loss function and its gradient. [20]

Odpověď (česky):

Cíl: Natrénovat binární logistickou regresi minimalizací *regularizované* ztrátové funkce.

1. Ztrátová funkce (Logistická regrese + L^2 penalizace) Pro *binární* klasifikaci (třídy např. C_1 a C_2) použijeme:

$$E(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}\Big(-\log\, p(C_{t_i} \mid \mathbf{x}_i; \mathbf{w})\Big) \;+\; \frac{\lambda}{2}\,\|\mathbf{w}\|^2,$$

kde $\lambda \geq 0$ je regulační parametr a $\mathbf{w}$ vektor vah. Pravděpodobnost $p(C_{t_i} \mid \mathbf{x}_i; \mathbf{w})$ je určena *logistickou funkcí* (sigmoid):

$$\sigma(z) \;=\; \frac{1}{1+e^{-z}}, \quad z = \mathbf{x}_i^T\mathbf{w}.$$

Pro binární výstup $t_i \in \{0, 1\}$ pak

$$p(C_{t_i} \mid \mathbf{x}_i; \mathbf{w}) = \sigma\big(\mathbf{x}_i^T\mathbf{w}\big)^{\,t_i}\;\big(1-\sigma(\mathbf{x}_i^T\mathbf{w})\big)^{\,1-t_i},$$

ale častěji se přímo používá

$$-\log\, p(C_{t_i} \mid \mathbf{x}_i; \mathbf{w}) = -\Big[t_i \log\sigma(\mathbf{x}_i^T\mathbf{w}) \;+\; (1-t_i)\,\log\big(1-\sigma(\mathbf{x}_i^T\mathbf{w})\big)\Big].$$

2. Gradient ztráty Pokud označíme $\hat{y}_i = \sigma(\mathbf{x}_i^T\mathbf{w})$, potom:

$$\nabla_{\mathbf{w}}\left[-\log\, p(t_i \mid \mathbf{x}_i; \mathbf{w})\right] \;=\; \big(\hat{y}_i - t_i\big)\,\mathbf{x}_i.$$

Pro *celkovou* (průměrnou) ztrátu a navíc s L^2 regulací $\frac{\lambda}{2}\,\|\mathbf{w}\|^2$, dostaneme:

$$\nabla_{\mathbf{w}}\, E(\mathbf{w}) \;=\; \frac{1}{N}\sum_{i=1}^{N}\big(\sigma(\mathbf{x}_i^T\mathbf{w}) - t_i\big)\,\mathbf{x}_i \;+\; \lambda\,\mathbf{w}.$$

Tento vzorec je dostačující k implementaci v `NumPy` (nebo jiném frameworku).

3. Minibatch SGD algoritmus

1. **Inicializace vah $\mathbf{w}$** (např. náhodně nebo nulovým vektorem).
2. **Opakuj (až do konvergence nebo max. počtu epoch):**
   - *Vyber minibatch* $\mathcal{B}$ z trénovacích příkladů $\{(\mathbf{x}_i, t_i)\}$.
   - *Spočti gradient* pro minibatch:

     $$\mathbf{g} \;=\; \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\Big(\sigma(\mathbf{x}_i^T\mathbf{w}) - t_i\Big)\,\mathbf{x}_i \;+\; \lambda\,\mathbf{w}.$$

   - *Aktualizuj váhy*:

     $$\mathbf{w} \;\leftarrow\; \mathbf{w} - \alpha\,\mathbf{g},$$

     kde α je učicí rychlost (learning rate).

4. Interpretace

- **Sigmoid** převádí reálné číslo $\mathbf{x}^T\mathbf{w}$ na interval $(0, 1)$, interpretovaný jako odhad $\hat{y}_i$ pravděpodobnosti třídy C_1.
- **Regularizace** $\frac{\lambda}{2}\|\mathbf{w}\|^2$ zabraňuje přespříliš velikým vahám a pomáhá zamezit overfittingu.
- **Minibatch** aktualizace zlepšuje efektivitu výpočtu i konvergenci u velkých datasetů (kompromis mezi Batch GD a čistě Stochastickým GD).

Tímto postupem získáme binární klasifikátor založený na logistické regresi, jenž se učí *metodou maximální věrohodnosti* (MLE) s negativním log-likelihoodem jako cílovou funkcí a s dodatečnou penalizací velikosti vah (L^2 regularizace).

Q3.7 Provide an intuitive justification for why cross-entropy is a good optimization objective in machine learning. What distributions do we compare in cross-entropy? Why is it good when the cross-entropy is low? [5]

Odpověď (česky):

Co v cross-entropii porovnáváme V rámci strojového učení je naším cílem *přiblížit* modelové rozdělení $p_{\text{model}}(x)$ (resp. $p_{\text{model}}(\text{třída} \mid x)$) k *empirickému* či „skutečnému“ rozdělení $\hat{p}_{\text{data}}(x)$ (resp. $\hat{p}_{\text{data}}(\text{třída} \mid x)$). Křížová entropie mezi nimi se typicky zapisuje jako

$$H\big(\hat{p}_{\text{data}}, p_{\text{model}}\big) = -\sum_x \hat{p}_{\text{data}}(x)\, \log p_{\text{model}}(x),$$

případně (v klasifikaci)

$$-\sum_{i=1}^{N}\sum_k y_{i,k}\, \log p_{\text{model}}(k \mid x_i),$$

kde $y_{i,k}$ je one-hot vektor s 1 u správné třídy a 0 u ostatních.

Proč je křížová entropie vhodná

- **Ztotožnění s NLL**: Minimalizací křížové entropie současně *minimalizujeme negativní log-likelihood*, což znamená, že model zvyšuje pravděpodobnost (podle p_{model}) skutečně pozorovaných dat (podle $\hat{p}_{\text{data}}$).
- **Vhodné pro klasifikaci**: V klasifikačních úlohách, kde „pravé“ rozdělení $\hat{p}_{\text{data}}$ často reprezentujeme *jednoznačně* (one-hot labely), cross-entropy přímo penalizuje případy, kdy model p_{model} dává malé pravděpodobnosti správným třídám.
- **Hladký gradient**: Cross-entropy (spolu se sigmoid nebo softmax výstupem) má dobře definované gradienty, které efektivně fungují s gradientními metodami a netrpí příliš tzv. mizejícím gradientem.
- **Interpretace**: Pokud se p_{model} a $\hat{p}_{\text{data}}$ „shodují“, je cross-entropy nízká. Naopak čím větší rozdíl mezi modelovým a skutečným rozdělením, tím vyšší hodnota.

Proč je dobré, když je křížová entropie nízká

- *Shoda modelu se skutečností*: Nízká křížová entropie znamená, že p_{model} se blíží „reálnému“ $\hat{p}_{\text{data}}$. V praxi se to projevuje menší chybovostí modelu (např. vyšší přesností klasifikace).
- *Maximalizace pravděpodobnosti*: Jelikož $H(\hat{p}_{\text{data}}, p_{\text{model}})$ lze chápat jako $-\sum \hat{p}_{\text{data}}(x) \log p_{\text{model}}(x)$, minimalizovat tuto veličinu znamená *maximalizovat* log-likelihood pro pozorovaná data.

Shrnutí: Křížová entropie je výhodná proto, že přímo měří „nesoulad“ mezi skutečnou distribucí ($\hat{p}_{\text{data}}$) a modelovou distribucí (p_{model}). Její minimalizace vede model k tomu, aby $\hat{p}_{\text{model}}$ co nejvěrněji odrážela reálné (empirické) rozdělení a tím zlepšila predikční výkon.

Part IV

Q4.1 Define mean squared error and show how it can be derived using MLE. What assumptions do we make during such derivation? [10]

Znění (v češtině):

Střední kvadratická chyba (MSE) je často používána jako ztrátová funkce pro regresní úlohy a lze ji odvodit z přístupu Maximální věrohodnosti (MLE), pokud předpokládáme, že cílová proměnná t, podmíněná na vstupu $\mathbf{x}$, je normálně rozložena s střední hodnotou rovnou výstupu modelu $y(\mathbf{x}; \mathbf{w})$ a s variancí σ^2. Za tohoto předpokladu je pravděpodobnostní rozdělení pro t dáno

$$p\big(t \mid \mathbf{x}; \mathbf{w}\big) = \mathcal{N}\big(t; y(\mathbf{x}; \mathbf{w}), \sigma^2\big).$$

Provádíme MLE a hledáme parametry $\mathbf{w}$, které maximalizují věrohodnost pozorovaných dat, což je ekvivalentní minimalizaci záporného log-likelihoodu. To vede k MSE následovně:

$$\mathbf{w}_{\text{MLE}} = \arg\max_{\mathbf{w}} \prod_{i=1}^{N} p\big(t_i \mid \mathbf{x}_i; \mathbf{w}\big) = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} -\log p\big(t_i \mid \mathbf{x}_i; \mathbf{w}\big).$$

Po dosazení normálního rozdělení a zanedbání konstant (nezávislých na $\mathbf{w}$) získáme

$$\arg\min_{\mathbf{w}} \sum_{i=1}^{N} (y(\mathbf{x}_i; \mathbf{w}) - t_i)^2.$$

A jelikož MSE je pouze tato suma vydělená počtem vzorků N, dojdeme ke tvaru:

$$E(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \big(y(\mathbf{x}_i; \mathbf{w}) - t_i\big)^2,$$

což je známá podoba MSE. Tím se ukazuje, že při *normálním* předpokladu pro chybu modelu vede MLE přirozeně k použití MSE jako minimalizované ztráty.

Q4.2 Considering K-class logistic regression model, write down its parameters (including their size) and explain how we decide what classes the input data belong to (including the formula for the softmax function). [10]

Odpověď (česky):

Parametry K-třídní logistické regrese

- Váhová matice $W \in \mathbb{R}^{D \times K}$ (každý sloupec odpovídá jedné třídě).
- Bias $\mathbf{b} \in \mathbb{R}^K$ (pro každou třídu jeden skalár).

Vstup $\mathbf{x} \in \mathbb{R}^D$. Číslo tříd je K.

Rozhodovací pravidlo pomocí softmax Nejprve spočteme $\mathbf{z} = \mathbf{x^T W} + \mathbf{b}$, přičemž $\mathbf{z}$ má délku K. Poté aplikujeme *softmax*:

$$\text{softmax}(\mathbf{z})_k = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}, \quad k = 1, \ldots, K.$$

Výstupem je vektor pravděpodobností $\big[p_1, p_2, \ldots, p_K\big]$. V praxi přiřadíme vstup $\mathbf{x}$ do třídy:

$$\hat{y} = \arg\max_k \text{softmax}(\mathbf{z})_k.$$

Q4.3 Explain the relationship between the sigmoid function and softmax. [5]

Odpověď (česky):

- **Sigmoid** $\sigma(z) = \frac{1}{1+e^{-z}}$ je speciální případ softmaxu pro *dvě* třídy ($K = 2$).
- Když máme dvě třídy, můžeme vektoru $\mathbf{z} = (z_1, z_2)$ aplikovat softmax:

  $$p_1 = \frac{\exp(z_1)}{\exp(z_1) + \exp(z_2)}, \quad p_2 = \frac{\exp(z_2)}{\exp(z_1) + \exp(z_2)}.$$

  Pokud nastavíme $z_2 = 0$, získáme $p_1 = \sigma(z_1)$ a $p_2 = 1 - \sigma(z_1)$.
- **Obecně** pro $K > 2$ se používá softmax, pro $K = 2$ stačí sigmoid.

Q4.4 Show that the softmax function is invariant towards constant shift. [5]

Odpověď (česky):

Nechť $\mathbf{z} = (z_1, \ldots, z_K)$ a definujme softmax:

$$\text{softmax}(z)_k = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}.$$

Pro libovolnou konstantu c uvažujme $\mathbf{z}' = (z_1 + c, \ldots, z_K + c)$. Pak

$$\text{softmax}(\mathbf{z}')_k = \frac{\exp(z_k + c)}{\sum_{j=1}^{K} \exp(z_j + c)} = \frac{\exp(c)\exp(z_k)}{\sum_{j=1}^{K} \exp(c)\exp(z_j)} = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)} = \text{softmax}(\mathbf{z})_k.$$

Tedy přičtení konstanty k všem složkám $\mathbf{z}$ *nemění* výsledek softmaxu.

Q4.5 Write down an L^2-regularized minibatch SGD algorithm for training a K-class logistic regression model, including explicit formulas of the loss function and its gradient. [20]

Odpověď (česky):

1. Vstup a inicializace

- **Vstup:** $\mathbf{X} \in \mathbb{R}^{N \times D}$ (trénovací data), $\mathbf{t} \in \{0, 1, \ldots, K-1\}^N$ (cílové třídy), učicí rychlost $\alpha \in \mathbb{R}^+$, regulační parametr $\lambda \in \mathbb{R}^+$.
- **Parametry modelu:** $\mathbf{W} \in \mathbb{R}^{D \times K}$ (matice vah) a $\mathbf{b} \in \mathbb{R}^K$ (bias). Inicalizujeme je např. na malé náhodné hodnoty.

2. Ztrátová (loss) funkce Pro minibatch $\mathcal{B} \subset \{1, \ldots, N\}$ definujeme *regularizovanou* negativní log-likelihood ztrátu:

$$E(\mathbf{W}) \;=\; -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \log\big(p(C_{t_i} \mid \mathbf{x}_i; \mathbf{W})\big) \;+\; \frac{\lambda}{2} \|\mathbf{W}\|_F^2,$$

kde $\|\mathbf{W}\|_F^2$ je součet čtverců všech prvků $\mathbf{W}$. Pravděpodobnost pro třídu t_i určíme pomocí softmax:

$$p(C_{t_i} \mid \mathbf{x}_i; \mathbf{W}) \;=\; \text{softmax}(\mathbf{z}_i)_{t_i}, \quad \text{kde } \mathbf{z}_i = \mathbf{x}_i^T \mathbf{W} + \mathbf{b}.$$

3. Gradient ztráty pomocí one-hot vektoru Nechť $\mathbf{1}_{t_i} \in \{0,1\}^K$ je **one-hot** reprezentace třídy t_i (tj. 1 na pozici t_i, 0 jinde). Definujme

$$\mathbf{p}_i = \text{softmax}(\mathbf{z}_i) \;\in\; \mathbb{R}^K.$$

Pak (v jednovzorkovém vyjádření) můžeme psát gradient vůči $\mathbf{W}$:

$$\nabla_{\mathbf{W}} E(\mathbf{W}) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \big[\, (\mathbf{p}_i - \mathbf{1}_{t_i})\, \mathbf{x}_i^T \big]^T \;+\; \lambda\, \mathbf{W},$$

kde $\mathbf{p}_i - \mathbf{1}_{t_i}$ je vektor $(K \times 1)$ a $\mathbf{x}_i^T$ je $(1 \times D)$, takže výsledná matice je $(D \times K)$.

(Pozn.: v některých textech se uvádí $(\mathbf{p}_i - \mathbf{1}_{t_i})\,\mathbf{x}_i^T$ nebo $\mathbf{x}_i(\mathbf{p}_i - \mathbf{1}_{t_i})^T$, jde o ekvivalentní zápisy.)

Pro gradient vůči biasu $\mathbf{b}$ (rozměru K) odpadá násobení $\mathbf{x}_i$:

$$\nabla_{\mathbf{b}} E(\mathbf{W}) \;=\; \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (\mathbf{p}_i - \mathbf{1}_{t_i}).$$

Bias $\mathbf{b}$ obvykle neregularizujeme.

4. Minibatch SGD algoritmus (L2 regularizace)

1. **Inicializace:** nastavit $\mathbf{W}, \mathbf{b}$.
2. **repeat (až do konvergence nebo max. počtu epoch):**
   - Zvol minibatch $\mathcal{B} \subset \{1, \ldots, N\}$.
   - Spočti gradient $\nabla_{\mathbf{W}} E_{\mathcal{B}}$, $\nabla_{\mathbf{b}} E_{\mathcal{B}}$ podle výše uvedených vzorců.
   - Aktualizuj:
   $$\mathbf{W} \leftarrow \mathbf{W} \;-\; \alpha\, \nabla_{\mathbf{W}} E_{\mathcal{B}}, \quad \mathbf{b} \leftarrow \mathbf{b} \;-\; \alpha\, \nabla_{\mathbf{b}} E_{\mathcal{B}}.$$

Shrnutí Takto definované gradienty (s softmax a one-hot $\mathbf{1}_{t_i}$) přesně odpovídají snímku, kde je gradient $\mathbf{y}(\mathbf{x}) - \mathbf{1}_t$ vynásoben vektorem $\mathbf{x}^\top$. L2 regulace $(\lambda \|\mathbf{W}\|^2)$ pak přidává $\lambda\,\mathbf{W}$ k tomuto gradientu. Tento postup minimalizuje *regularizovanou* zápornou log-likelihood, čímž model softmax-logistic regrese dosahuje dobré generalizace na K třídách.

Q4.6 Prove that decision regions of a multiclass logistic regression are convex. [10]

Znění (v češtině):

Abychom ukázali konvexnost rozhodovacích oblastí ve vícetřídní (multiclass) logistické regresi, uvažujme dva body $\mathbf{x}_A$ a $\mathbf{x}_B$ ze stejné rozhodovací oblasti R_k. Kriterium rozhodování ve vícetřídní logistické regresi je založeno na lineárních funkcích $\mathbf{x}^T W$, kde W je matice vah. Bod $\mathbf{x}$ patří do oblasti R_k právě tehdy, když

$$\hat{y}(\mathbf{x})_k = \left(\mathbf{x}^T W\right)_k$$

je největší ze všech třídních skóre. Pro body $\mathbf{x}_A, \mathbf{x}_B \in R_k$ a libovolné $\lambda \in [0, 1]$ ukážeme, že jejich konvexní kombinace $\mathbf{x} = \lambda \mathbf{x}_A + (1 - \lambda)\,\mathbf{x}_B$ splňuje

$$\hat{y}(\mathbf{x})_k = \lambda\, \hat{y}(\mathbf{x}_A)_k + (1 - \lambda)\, \hat{y}(\mathbf{x}_B)_k.$$

Protože $\hat{y}(\mathbf{x}_A)_k$ i $\hat{y}(\mathbf{x}_B)_k$ jsou největší skóre pro své body (v příslušné třídě k), stejné tvrzení platí i pro konvexní kombinaci $\mathbf{x}$, která tedy také spadá do R_k. To platí pro libovolnou konvexní kombinaci bodů v R_k, z čehož plyne, že R_k je konvexní množina.

Q4.7 Considering a single-layer MLP with D input neurons, H hidden neurons, K output neurons, hidden activation f, and output activation a, list its parameters (including their size) and write down how the output is computed. [10]

Odpověď (česky):

Parametry MLP (jednovrstvé, 1 hidden layer)

- Váhy a bias pro skrytou vrstvu:

$$W^{(h)} \in \mathbb{R}^{D \times H}, \quad \mathbf{b}^{(h)} \in \mathbb{R}^{H}.$$

- Váhy a bias pro výstupní (druhou) vrstvu:

$$W^{(y)} \in \mathbb{R}^{H \times K}, \quad \mathbf{b}^{(y)} \in \mathbb{R}^{K}.$$

Výpočet výstupu

$$\mathbf{h} = f\big(\mathbf{x}^T W^{(h)} + \mathbf{b}^{(h)}\big), \quad \hat{\mathbf{y}} = a\big(\mathbf{h}^T W^{(y)} + \mathbf{b}^{(y)}\big).$$

Zde:

- $\mathbf{x} \in \mathbb{R}^D$ je vstup.
- $\mathbf{h} \in \mathbb{R}^H$ je vektor skrytých aktivací pomocí funkce f.
- $\hat{\mathbf{y}} \in \mathbb{R}^K$ je výstup (před finální interpretací), pomocí aktivační funkce a.

Q4.8 List the definitions of frequently used MLP output layer activations (the ones producing parameters of a Bernoulli distribution and a categorical distribution). Then, write down three commonly used hidden layer activations (sigmoid, tanh, ReLU). Explain why identity is not a suitable activation for hidden layers. [10]

Odpověď (česky):

Výstupní aktivační funkce (Bernoulli, kategorie):

- **Sigmoid** $\sigma(z) = \frac{1}{1+e^{-z}}$ – pro *binární* výstup (pravděpodobnost třídy). Parametr Bernoulliho rozdělení.
- **Softmax** $\text{softmax}(\mathbf{z})_k = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$ – pro *multitřídní* výstup. Parametry kategorického rozdělení.

Skryté vrstvy: tři běžné aktivační funkce

- **Sigmoid**: $\sigma(z)$.
- **Tanh**: $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$.
- **ReLU**: $\text{ReLU}(z) = \max(0, z)$.

Proč *není vhodná identita* pro hidden vrstvy

- Bez nelineární aktivace se MLP *zredukuje* na lineární model (složené lineární transformace jsou opět lineární).
- Ztrácíme schopnost modelovat *nelineární* vztahy v datech.

Part V

Q5.1 Considering a single-layer MLP with D input neurons, a ReLU hidden layer with H units, and a softmax output layer with K units, write down the explicit formulas (i.e., formulas you would use to code in `numpy`) of the gradient of all the MLP parameters (two weight matrices and two bias vectors), assuming input x, target t, and negative log likelihood loss. [20]

Intuitivní vysvětlení

Vícevrstvý perceptron (MLP) je model, který se učí predikovat výstupy y na základě vstupů x přes jednu nebo více skrytých vrstev. Ztrátová funkce (loss) měří, jak dobře model funguje, a gradient ztrátové funkce podle vah a biasů nám říká, jak tyto parametry upravit, aby se zlepšila predikce. Používáme **řetízkové pravidlo**, abychom rozložili derivaci složené funkce zpětně od výstupu ke vstupu.

Zde je složená funkce:

$$L = -\log(\text{softmax}(z^{(y)})_k), \quad \text{kde} \quad z^{(y)} = h^T W^{(y)} + b^{(y)}, \quad h = \text{ReLU}(x^T W^{(h)} + b^{(h)}).$$

Nyní budeme derivovat gradienty podle vah $W^{(y)}, W^{(h)}$ a biasů $b^{(y)}, b^{(h)}$.

Výpočet gradientů

Gradient podle $W^{(y)}$ a $b^{(y)}$ (výstupní vrstva)

1. Derivace ztráty podle výstupů softmax (zápis odpovídá kódu v `numpy`):

$$\frac{\partial L}{\partial z^{(y)}} = \text{softmax}(z^{(y)}) - t.$$

   Tato derivace vzniká díky vlastnostem softmax funkce a její kombinaci s negativní logaritmickou pravděpodobností. Vektor pravděpodobností je snížen o vektor skutečných hodnot (targetů).

2. Gradient ztráty podle vah $W^{(y)}$:

$$\frac{\partial L}{\partial W^{(y)}} = h^T \cdot \frac{\partial L}{\partial z^{(y)}}.$$

   Toto vyplývá z faktu, že $z^{(y)} = h^T W^{(y)} + b^{(y)}$, takže derivace podle $W^{(y)}$ je vynásobení výstupu skryté vrstvy h a gradientu loss funkce podle $z^{(y)}$.

3. Gradient ztráty podle biasů $b^{(y)}$:

$$\frac{\partial L}{\partial b^{(y)}} = \sum_j \frac{\partial L}{\partial z_j^{(y)}}.$$

Bias ovlivňuje všechny jednotky přímo, takže jeho gradient je součet přes všechny výstupy softmaxu.

Gradient podle $W^{(h)}$ a $b^{(h)}$ (skrytá vrstva)

1. Gradient ztráty podle výstupů skryté vrstvy h:

$$\frac{\partial L}{\partial h} = \frac{\partial L}{\partial z^{(y)}} \cdot W^{(y)}.$$

Toto následuje z toho, že h ovlivňuje $z^{(y)}$ skrze násobení vahami $W^{(y)}$.

2. Gradient ztráty podle vstupů do skryté vrstvy $z^{(h)}$ (ReLU derivace):

$$\frac{\partial L}{\partial z_{jk}^{(h)}} = \begin{cases} \frac{\partial L}{\partial h_{jk}}, & \text{pokud } h_{jk} > 0, \\ 0, & \text{jinak.} \end{cases}$$

ReLU funkce je nelineární, a proto její derivace je 1 pro pozitivní hodnoty a 0 pro negativní.

3. Gradient ztráty podle vah $W^{(h)}$:

$$\frac{\partial L}{\partial W^{(h)}} = x^T \cdot \frac{\partial L}{\partial z^{(h)}}.$$

Gradient podle vah první vrstvy $W^{(h)}$ vyplývá z toho, že $z^{(h)} = x^T W^{(h)} + b^{(h)}$, a je tedy ovlivněn vstupy x.

4. Gradient ztráty podle biasů $b^{(h)}$:

$$\frac{\partial L}{\partial b^{(h)}} = \sum_j \frac{\partial L}{\partial z_j^{(h)}}.$$

Bias v první vrstvě ovlivňuje všechny jednotky skryté vrstvy přímo.

Q5.2 Formulate the computation of MLP as a computation graph. Explain how such a graph can be used to compute the gradients of the parameters in the back-propagation algorithm. [10]

Intuitivní vysvětlení

Výpočetní graf reprezentuje jednotlivé kroky dopředné propagace MLP. Uzel v grafu představuje výpočet (např. násobení, sčítání nebo aplikaci aktivace). Hrany představují datové toky mezi výpočty. Při zpětné propagaci derivujeme každý uzel postupně pomocí **řetízkového pravidla**.

Výpočetní graf

- **Vstupní vrstva**: x vstupuje do výpočtu $z^{(h)} = x^T W^{(h)} + b^{(h)}$.
- **Skrytá vrstva**: $h = \text{ReLU}(z^{(h)})$.
- **Výstupní vrstva**: $z^{(y)} = h^T W^{(y)} + b^{(y)}$, $y = \text{softmax}(z^{(y)})$.
- **Ztráta**: $L = -\log(y_i)_k$.

Použití grafu pro zpětnou propagaci

1. Začněte od ztráty L. Derivujte $\frac{\partial L}{\partial y}$ (výstup softmax).
2. Propagujte chybu zpět k $z^{(y)}$ a spočítejte gradienty pro $W^{(y)}$ a $b^{(y)}$.
3. Pokračujte k h, kde chybu propagujete přes ReLU na $z^{(h)}$.
4. Spočítejte gradienty pro $W^{(h)}$ a $b^{(h)}$.
5. Aktualizujte váhy a biasy podle gradientů.

Q5.3 Formulate the Universal Approximation Theorem and explain in words what it says about multi-layer perceptrons. [10]

Odpověď (česky):

Univerzální aproximační teorém (Hornik, Cybenko, 1989 apod.)

- Tvrzení: Nechť $\varphi : \mathbb{R} \to \mathbb{R}$ je nekonstantní, omezená a monotonní funkce (např. sigmoid, ReLU). Pak pro každou spojitou funkci $f : [0,1]^D \to \mathbb{R}$ a libovolné $\varepsilon > 0$ existuje jistý počet skrytých neuronů (H), váhový vektor $\mathbf{v}$ a matice $\mathbf{W}$, bias $\mathbf{b}$, takové, že aproximační síť

$$F(\mathbf{x}) = \mathbf{v}^T \varphi\big(\mathbf{x}^T \mathbf{W} + \mathbf{b}\big)$$

  se od $f(\mathbf{x})$ liší o méně než ε pro všechny $\mathbf{x}$ v dané doméně.

Význam pro MLP:

- Jednovrstvý (s jednou skrytou vrstvou) *vícevrstvý perceptron* s vhodným (dostatečně velkým) počtem skrytých neuronů *dokáže approximovat* libovolnou spojitou funkci na kompaktní množině $\subseteq \mathbb{R}^D$.
- Teorém zaručuje *existenci* sady vah a biasů, ovšem *negarantuje*, že učení vždy najde tuto optimální (či téměř optimální) konfiguraci.
- Teorém se týká *reprezentační* schopnosti MLP, nikoli rychlosti či snadnosti trénování.

Q5.4 How do we search for a minimum of a function $f(x) : \mathbb{R}^D \to \mathbb{R}$ subject to equality constraints $g_1(x) = 0, \ldots, g_m(x) = 0$? [10]

Odpověď (česky):

Vyhledávání minima s vazbami (metoda Lagrangeových multiplikátorů)

- Chceme minimalizovat $f(\mathbf{x})$ za podmínek $\{g_1(\mathbf{x}) = 0, \ldots, g_m(\mathbf{x}) = 0\}$.
- Zavedeme *Lagrangeovu funkci*:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i \, g_i(\mathbf{x}),$$

  kde $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$ jsou tzv. Lagrangeovy multiplikátory.

- Podmínka stacionarity:

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = 0, \quad \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = 0.$$

  – První rovnost říká, že gradient $\mathbf{x}$-ové složky Lagrangiánu je nulový:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i \nabla_{\mathbf{x}} g_i(\mathbf{x}) = 0.$$

  – Druhá rovnost odpovídá zapracování vazeb do kritéria, tedy $g_i(\mathbf{x}) = 0$ pro všechna i.

- Řešením těchto rovnic spolu s podmínkami $g_i(\mathbf{x}) = 0$ dostáváme kandidátní body minima (případně maxima nebo sedla — je potřeba ověřit, zda skutečně jde o minimum).

Q5.5 Prove which categorical distribution with N classes has maximum entropy. [10]

Odpověď (česky):

Maximalizace entropie pro kategorické rozdělení

- Uvažujme $\mathbf{p} = (p_1, \ldots, p_N)$ takové, že $p_i \geq 0$ a $\sum_{i=1}^{N} p_i = 1$.
- Entropie je

$$H(\mathbf{p}) = -\sum_{i=1}^{N} p_i \, \log p_i.$$

- Chceme $\mathbf{p}$ s *maximální* entropií, podmíněné $\sum_i p_i = 1$.
- Převedeme problém na minimalizaci záporné entropie $-H(\mathbf{p})$

Lagrangeova funkce

$$\mathcal{L}(\mathbf{p}, \lambda) = \sum_{i=1}^{N} p_i \log p_i - \lambda\Big(\sum_{i=1}^{N} p_i - 1\Big).$$

- Derivací podle p_i a položením rovné nule:

$$0 = \frac{\partial \mathcal{L}}{\partial p_i} = 1 \log p_i + p_i \frac{1}{p_i} - \lambda \quad \Longrightarrow \quad p_i = e^{\lambda - 1}.$$

- Z podmínky $\sum_i p_i = 1$ plyne:

$$\sum_{i=1}^{N} e^{\lambda-1} = N\, e^{\lambda-1} = 1 \;\; \Longrightarrow \;\; e^{\lambda-1} = \frac{1}{N}.$$

- Tím vychází $p_i = \frac{1}{N}$. To je *rovnoměrné* rozdělení.

Závěr: Rozdělení s největší entropií (mezi kategoriemi) je uniformní $\left(\frac{1}{N}, \ldots, \frac{1}{N}\right)$.

Q5.6 Consider derivation of softmax using maximum entropy principle, assuming we have a dataset of N examples (x_i, t_i), $x_i \in \mathbb{R}^D, t_i \in \{1, 2, \ldots, K\}$. Formulate the three conditions we impose on the searched $\pi : \mathbb{R}^D \to \mathbb{R}^K$, and write down the Lagrangian to be minimized. [20]

Odpověď (česky):

Motivace: maximum entropy pro softmax Hledáme funkci $\pi(x) \in \mathbb{R}^K$, která pro daný vstup x vrací pravděpodobnostní rozdělení tříd. Budeme chtít minimalizovat $-\sum_{i=1}^{N} H(\pi(x_i))$. Funkce $\pi(x)$ musí být co nejméně "zaujatá"(proto maximalizace entropie) a zároveň musí splňovat následující podmínky:

1. $\pi(x)_k \geq 0 \quad \forall k, x$. Pravděpodobnosti musí být nezáporné.
2. $\sum_{k=1}^{K} \pi(x)_k = 1 \quad \forall x$. Součet pravděpodobností přes všechny třídy musí být roven 1.
3. $\sum_{i=1}^{N} \pi(x_i)_k x_i = \sum_{i=1}^{N} \mathbf{1}[t_i = k] x_i \quad \forall k$. Vážený průměr vstupů x_i podle pravděpodobností $\pi(x_i)_k$ musí odpovídat průměru vstupů, které skutečně patří do třídy k (empirické distribuci).

Lagrangeova funkce Pro minimalizaci záporné *entropie* (maximalizaci kladné entropie) $\sum_{i=1}^{N} \sum_{k=1}^{K} \pi(x_i)_k \log \pi(x_i)_k$ při dodržení těchto podmínek definujeme Lagrangián:

$$\mathcal{L} = \sum_{i=1}^{N} \sum_{k=1}^{K} \pi(x_i)_k \log \pi(x_i)_k - \sum_{i=1}^{N} \beta_i \Big(\sum_{k=1}^{K} \pi(x_i)_k - 1\Big) - \sum_{j=1}^{D} \sum_{k=1}^{K} \lambda_{j,k} \Big(\sum_{i=1}^{N} \pi(x_i)_k x_{i,j} - \sum_{i=1}^{N} \mathbf{1}[t_i = k] x_{i,j}\Big),$$

kde:

- $\sum_{i=1}^{N}\sum_{k=1}^{K}\pi(x_i)_k \log \pi(x_i)_k$ je negativní entropie, kterou chceme minimalizovat,
- $\sum_{i=1}^{N}\beta_i\Big(\sum_{k=1}^{K}\pi(x_i)_k - 1\Big)$ odpovídá podmínce normalizace (součet pravděpodobností je 1),
- $\sum_{j=1}^{D}\sum_{k=1}^{K}\lambda_{j,k}\Big(\sum_{i=1}^{N}\pi(x_i)_k x_{i,j} - \sum_{i=1}^{N}\mathbf{1}[t_i = k]x_{i,j}\Big)$ vyjadřuje podmínku shody váženého průměru modelu s empirickými daty.

Shrnutí Definované podmínky zajišťují:

1. Nezápornost pravděpodobností ($\pi(x)_k \geq 0$).
2. Normalizaci pravděpodobnostního rozdělení ($\sum_{k=1}^{K}\pi(x)_k = 1$).
3. Shodu váženého průměru vstupů s empirickými daty (x_i vázané na třídu k).

Lagrangeova funkce umožňuje kombinovat tyto podmínky s maximalizací entropie, což vede k optimálnímu řešení, které má tvar $\text{softmax}(\mathbf{x}_k^T w + b_k)$.

Q5.7 Define precision (including true positives and others), recall, F_1 score, and F_β score (we stated several formulations for F_1 and F_β scores; any one of them will do). [10]

Odpověď (česky):

Základ: Confusion Matrix (matice záměn)

- **True Positives (TP):** Případy, které jsou skutečně pozitivní a model je správně označil za pozitivní.
- **False Positives (FP):** Skutečně negativní případy, které model chybně označil za pozitivní.
- **True Negatives (TN):** Skutečně negativní případy, které model správně označil za negativní.
- **False Negatives (FN):** Skutečně pozitivní případy, které model označil za negativní.

Precision (presnost)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Vyjadřuje, jaký podíl označených (predikovaných) pozitiv je *skutečně* pozitivních.

Recall (citlivost)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Udává, jak velký podíl *skutečných* pozitiv model správně identifikuje.

F_1 skóre

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Je harmonickým průměrem presnosti a citlivosti (recall).

F_β skóre

$$F_\beta = (1 + \beta^2) \frac{\text{Precision} \times \text{Recall}}{\beta^2 \,\text{Precision} + \text{Recall}}.$$

Obecnější metrika, která dává větší váhu buď recall ($\beta > 1$) nebo precision ($\beta < 1$) podle zvoleného parametru β.

Q5.8 Explain the difference between micro-averaged and macro-averaged F_1 scores. [10]

Odpověď (česky):

Micro-averaged F_1

- V *micro průměru* se všechny třídy hromadně posuzují *dohromady*.
- Počítá se TP, FP, FN pro všechny třídy *společně* a poté se z nich vytváří $\text{Precision}_{\text{micro}}$ a $\text{Recall}_{\text{micro}}$.
- Vzorec například:

$$F_1^{\text{micro}} = 2 \times \frac{\text{Precision}_{\text{micro}} \times \text{Recall}_{\text{micro}}}{\text{Precision}_{\text{micro}} + \text{Recall}_{\text{micro}}} = \frac{2 \sum \text{TP}}{2 \sum \text{TP} + \sum \text{FP} + \sum \text{FN}}.$$

- Takto jsou chyby všech tříd shrnuty jako celek, hodí se tehdy, když chceme zohlednit *velikosti* tříd a ne penalizovat malé/velké třídy zvlášť.

Macro-averaged F_1

- *Makro průměr* nejprve spočte F_1 pro každou třídu zvlášť a poté vezme *prostý průměr* těchto výsledků.

$$F_1^{\text{macro}} = \frac{1}{K} \sum_{k=1}^{K} F_{1,k},$$

  kde $F_{1,k}$ je F_1 skóre pro třídu k.

- Vhodné tehdy, když každou třídu hodnotíme *stejně důležitě*, i když jsou třídy nevyvážené (v malých třídách je i malé množství vzorků).

Rozdíl: Micro-averaged metrika klade důraz na *celkový* počet chyb, zatímco macro-averaged metrika dává všem třídám *stejnou váhu* bez ohledu na jejich velikost.

Q5.9 Explain (using examples) why accuracy is not a suitable metric for unbalanced target classes, e.g., for a diagnostic test for a contagious disease. [5]

Odpověď (česky):

Nevhodnost accuracy pro nevyvážené (unbalanced) třídy

- Accuracy $= \frac{\text{TP+TN}}{\text{počet všech vzorků}}$.
- Pokud je jedna třída *vzácná* (např. nemoc), model může triválně predikovat „nemoc = ne“ (vše negativní) a získat vysokou přesnost.
- V příkladu se 1000 jedinci, z nichž má jen 10 nemoc:

$$\text{Accuracy} = \frac{0 + 990}{1000} = 99\% \quad \text{(všichni označeni za zdravé)}.$$

  Přitom model vůbec nemocné neodhalí (TP = 0).

Závěr: V nevyvážených úlohách (např. vzácné onemocnění) je accuracy *zrádná*. Ukazuje dobrá procenta úspěchu, ale nereflektuje fatální neschopnost modelu odhalit menšinovou třídu. Proto se používají vhodnější metriky (precision, recall, F_1 apod.), které lépe postihují výkon modelu na vzácné třídě.

Part VI

Q6.1 Explain how the TF-IDF weight of a given document-term pair is computed. [5]

Odpověď (česky):

Základní definice: TF-IDF pro dvojici (t, d) (term t, document d) se vypočítá jako:

$$\text{TF-IDF}(t, d) \;=\; \text{TF}(t, d) \;\times\; \text{IDF}(t, D),$$

kde

- $\text{TF}(t, d)$ je *term frequency*, tedy četnost výskytu termu t v dokumentu d (může být normalizovaná či nelineární, např. log-TF),
- $\text{IDF}(t, D)$ je *inverse document frequency* ve *korpusu* D (množina všech dokumentů).

Výpočet IDF:

$$\text{IDF}(t, D) \;=\; \log\left(\frac{|D|}{|\{\, d \in D : t \in d\}| \text{ (případně } +1)}\right),$$

kde

- $\{\, d \in D : t \in d\}$ je množina dokumentů obsahujících term t.

Intuice TF-IDF:

- *Term frequency (TF)* zvyšuje váhu termu úměrně tomu, jak často se v dokumentu objevuje.
- *Inverse document frequency (IDF)* tuto váhu *snižuje*, pokud je term obecně velmi častý v celém korpusu (jelikož by pak nebyl moc „relevantní“).
- Výsledkem je, že TF-IDF zvýhodňuje slova unikátní pro daný dokument oproti těm, jež se vyskytují plošně napříč všemi dokumenty.

Q6.2 Define conditional entropy, mutual information, write down the relation between them, and finally prove that mutual information is zero if and only if the two random variables are independent (you do not need to prove statements about D_{KL}). [10]

Odpověď (česky):

Podmíněná entropie $H(Y \mid X)$ Udává očekávanou hodnotu entropie náhodné proměnné Y, známe-li hodnotu X.

$$H(Y \mid X) = -\sum_{x \in X} \sum_{y \in Y} P(x, y) \log P(y \mid x).$$

Vzájemná informace $I(X;Y)$ Měří, *kolik informací* (v bitovém smyslu) o Y obsahuje X. Definice:

$$I(X;Y) = \sum_{x,y} P(x, y) \log \frac{P(x, y)}{P(x)\, P(y)}.$$

Vztah mezi $H(Y \mid X)$ **a** $I(X;Y)$

$$I(X;Y) = H(Y) - H(Y \mid X).$$

To znamená, že vzájemná informace je *zmenšení nejistoty* v Y díky znalosti X. Dále je $I(X;Y) = I(Y;X)$, tedy symetrická.

Důkaz: $I(X;Y) = 0 \Leftrightarrow X$ **a** Y **jsou nezávislé**

- Z definice $I(X;Y) = \sum_{x,y} P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$.
- Pokud jsou X a Y nezávislé, pak $P(x,y) = P(x)P(y)$, takže $\log \frac{P(x,y)}{P(x)P(y)} = 0$ a každý sčítanec je nulový, tudíž $I(X;Y) = 0$.
- Naopak pokud $I(X;Y) = 0$, znamená to $\log \frac{P(x,y)}{P(x)P(y)} = 0$ skoro všude (krom bodů s nulovou pravděpodobností), tedy $P(x,y) = P(x)P(y)$. Tím je prokázána nezávislost.

Q6.3 Show that TF-IDF terms can be considered portions of suitable mutual information. [10]

Odpověď (česky):

Interpretace TF-IDF jako vzájemné informace: Mějme kolekci dokumentů D a množinu termů T. Mutual information $I(D;T)$ může být interpretována jako informace sdílená mezi dokumenty a termy. Výpočet vzájemné informace lze provést následovně:

- Pravděpodobnost výběru dokumentu d z kolekce D je rovnoměrná:

$$P(d) = \frac{1}{|D|}, \quad I(d) = H(D) = \log |D|.$$

- Pravděpodobnost, že term t je obsažen v dokumentu d, je:

$$P(t \in d) = \frac{1}{|\{d \in D : t \in d\}|}.$$

- Informace o dokumentu vzhledem k termu t (podmíněná entropie) je definována jako:

$$I(d \mid t \in d) = H(D \mid t) = \log |\{d \in D : t \in d\}|.$$

- Rozdíl $I(d) - I(d \mid t \in d)$ odpovídá:

$$I(d) - I(d \mid t \in d) = \log |D| - \log |\{d \in D : t \in d\}| = \log \frac{|D|}{|\{d \in D : t \in d\}|} = \mathrm{IDF}(t).$$

Výpočet vzájemné informace: Vzájemná informace mezi D a T se pak spočítá jako:

$$I(D;T) = \sum_{d,t \in d} P(d) \cdot P(t \mid d) \cdot \big(I(d) - I(d \mid t)\big).$$

Dosazením výše uvedených vztahů pro $P(d)$, $P(t \mid d)$, a $I(d) - I(d \mid t)$ získáme:

$$I(D;T) = \frac{1}{|D|} \sum_{d,t \in d} \mathrm{TF}(t,d) \cdot \mathrm{IDF}(t),$$

kde:

- $\mathrm{TF}(t,d)$ je term frequency,
- $\mathrm{IDF}(t)$ je inverse document frequency.

Shrnutí: Každý TF-IDF vážený term přispívá k celkové vzájemné informaci mezi kolekcí dokumentů D a množinou termů T. Lze tedy říci, že TF-IDF váha představuje „dílčí část informace“ spojenou s dvojicí dokument-term.

Q6.4 Explain the concept of word embedding in the context of MLP and how it relates to representation learning. [5]

Odpověď (česky):

Word embedding a MLP:

- Word embedding je technika *reprezentačního učení*, kde slova z *slovníku* zobrazujeme do (typicky) hustých vektorů reálných čísel.
- Tyto vektory zachycují *významovou* podobnost slov, protože podobná slova se v embeddingovém prostoru nacházejí blízko sebe.
- V MLP se embeddingy využívají např. na vstupní vrstvě: místo jednorozměrných one-hot vektorů s tisíci rozměry tak používáme naučené nižší dimenze, jež MLP během tréninku dále *zdokonaluje* (fine-tuning).

Vztah k representation learning:

- Embeddingy umožňují MLP automaticky si osvojit různé jazykové nuance přímo z dat, bez nutnosti ručního feature engineeringu.
- Síť se tak naučí kódovat *syntaktické i sémantické* vztahy mezi slovy, což zvyšuje výkon v NLP úlohách (sentiment analýza, strojový překlad apod.).
- Word embeddings se dále rozvíjejí v pokročilejších modelech (RNN, LSTM, Transformer), které posouvají state-of-the-art v NLP.

Q6.5 Describe the skip-gram model trained using negative sampling. [10]

Odpověď (česky):

Princip skip-gram modelu

- Cílem je naučit *embeddingy slov* tak, aby dokázaly předpovídat *kontextová* slova (okolní slova) pro dané *cílové* (target) slovo.
- Klasický skip-gram by se snažil predikovat pro každé cílové slovo, která slova v celém slovníku *patří* do kontextu. To je však výpočetně nákladné.

Negative sampling (SGNS)

- Namísto explicitního modelování přítomnosti všech možných slov v kontextu se *SGNS* zaměřuje na *rozlišení* **skutečných** kontextových slov od **náhodně vybraných** (tzv. *negativních*) slov.
- Pro dvojici (w, c) (cílové slovo w a kontextové slovo c) se model učí *vysokou* pravděpodobnost (σ-aktivace) pro správný pár:
$$\log \sigma\big(\mathbf{v}_c^\top \mathbf{e}_w\big),$$
kde $\mathbf{e}_w$ je embedding cílového slova w a $\mathbf{v}_c$ je embedding kontextu c.
- **Negativní vzorky** se zpracují tak, že model *minimalizuje* pravděpodobnost (respektive $\log \sigma(\cdot)$ z opačného znaménka) pro dvojici (w, c_i), kde c_i je „náhodné slovo“, nemající být v kontextu:
$$-\log \sigma\big(-\mathbf{v}_{c_i}^\top \mathbf{e}_w\big).$$
- Celková objektivní funkce *skládá* kladné (správné) páry a záporné (negativní) páry, přičemž negativní slova se obvykle vzorkují na základě *frekvence* (noise distribution).

Výhoda

- SGNS dramaticky *snižuje* výpočetní náklady oproti verzi, kde by se vyhodnocoval *celý* slovník.
- V praxi lze díky negative sampling model trénovat i na velmi rozsáhlých korpusech a embeddingy se naučí kvalitní distribučně-sémantické reprezentace.

Q6.6 How would you proceed to train a part-of-speech tagger (i.e., you want to assign each word with its part of speech) if you only could use pre-trained word embeddings and MLP classifier? [5]

Odpověď (česky):

Zpracování dat

- Získat dataset s texty, kde jsou slova označena POS tagy (např. Universal Dependencies).
- *Tokenizovat* slova a nahradit je *předtrénovanými embeddingy* (např. GloVe, fastText, word2vec).
- Volitelně přidat kontextová vstupní data (např. i okolní embeddingy), dle potřeby.
- Cílové POS tagy zakódovat např. one-hot vektory (nebo labely 0,1,2...).

MLP architektura

- **Vstupní vrstva:** Vektor embeddingu (příp. konkatenace okolních embeddingů).
- **Skryté vrstvy:** Jedna či více vrstvy s nelineární aktivací (např. ReLU).
- **Výstupní vrstva:** softmax přes neurony, z nichž každý odpovídá jedné POS kategorii.

Trénovací proces

- Rozdělit dataset na trénovací, validační a testovací část.
- Trénovat MLP minimalizací cross-entropy (kategorizace) s typickými optimalizéry (Adam, SGD).
- Využít validační sadu pro ladění hyperparametrů (počet vrstev, velikost, α, λ) a *early stopping*.

Hodnocení a post-processing

- Vyhodnotit přesnost a F_1 skóre na testu.
- V případě sekvenční povahy (kde i sousední POS tagy hrají roli) lze doplnit *sekvenční* model (např. CRF) nad výstupy MLP.

Q6.7 What is Zipf's law? Explain how it can be used to provide intuitive justification for using the logarithm when computing IDF. [5]

Odpověď (česky):

Zipfův zákon (Zipf's law)

- Empirické zjištění z analýzy přirozených jazyků (a dalších oblastí), že frekvence výskytu slov klesá *přibližně* podle inverzní úměry k jejich pořadí v žebříčku nejčastějších slov.
- Formálně: pokud r je *rank* (pořadí) slova, potom frekvence f se často chová jako $f \sim \frac{1}{r^\alpha}$ (typicky $\alpha \approx 1$).
- Důsledek: velmi malá množina *nejfrekventovanějších* slov tvoří většinu všech výskytů, zatímco spousta slov se vyskytuje jen zřídka.

Vazba na použití logaritmu v IDF

- IDF (Inverse Document Frequency) se běžně definuje s logaritmem:

  $$\mathrm{IDF}(t) \;=\; \log\!\Big(\frac{N}{\mathrm{df}_t}\Big) \quad \text{(případně s } +1\text{, atd.)},$$

  kde N je počet dokumentů a df_t je počet dokumentů obsahujících term t.
- Podle Zipfova zákona existují extrémně častá slova (high frequency) i extrémně vzácná. Použití log zajišťuje, že *nesrazíme* IDF příliš tvrdě u velmi častých termů (a nevyzdvihneme *příliš* extrémně ty vzácné).
- Logaritmické měřítko *zjemňuje* rozdíly v extrémech: jedním slovem, log roste pomalu, takže skok mezi 10krát častějším slovem a 100krát častějším slovem není tak drastický v IDF jako kdybychom použili přímou úměrnost.

Intuitivní shrnutí: Zipfův zákon vysvětluje, že slova s nejvyšším rankem jsou v datech extrémně „přesycená". Použití log v IDF pomáhá kompenzovat tyto enormní rozdíly a „zmírnit" sílu penalizace pro běžná slova (a vyzdvižení pro ta vzácná), čímž IDF lépe reaguje na reálné rozložení frekvencí vyplývající z Zipfova zákona.

Part VII

Q7.1 Describe k-nearest neighbors prediction, both for regression and classification. Define L_p norm and describe uniform, inverse, and softmax weighting. [10]

Odpověď (česky):

1. Základ k-NN

- *k-nejbližších sousedů* (k-NN) využívá trénovací data $(\mathbf{x}_i, t_i)$, kde $\mathbf{x}_i$ je vstup (vektorem příznaků) a t_i je cíl (číselný pro regresi, kategorický pro klasifikaci).
- Pro nový bod $\mathbf{x}$ najdeme *k nejbližších* sousedů v trénovací sadě (podle nějaké metriky).
- *Predikce:* použijeme cíle těchto sousedů (tj. $\{t_{i_1}, t_{i_2}, \dots\}$) a „zprůměrujeme“ je vhodnou vahou (viz níže).

2. k-NN pro regresi Predikovaná hodnota cíle je vážený průměr cílů $\{t_i\}$:

$$\hat{t} = \frac{\sum_i w_i \, t_i}{\sum_i w_i}.$$

Pokud jsou váhy w_i všechny stejné, je to prostý průměr. Nebo lze vážit vzdáleností.

3. k-NN pro klasifikaci

- Varianta *Uniform weighting*: predikce je mode$\{t_1, t_2, \dots, t_k\}$, tedy nejčastější třída ze sousedů.
- Varianta s *neuniformními* váhami: volíme takovou třídu c, pro niž je $\sum_i w_i \, \mathbf{1}[t_i = c]$ maximální (nebo, ekvivalentně, sčítáme vážené indicie).

4. L_p-norma (metrika)

$$\|\mathbf{x} - \mathbf{y}\|_p = \left(\sum_{j=1}^{D} | \, x_j - y_j \, |^p \right)^{\frac{1}{p}}.$$

- Nejčastěji $p = 2$ (Euklidovská), $p = 1$ (Manhattan).

5. Váhovací metody

- **Uniform**: $w_i = 1$ (všichni sousedé stejná váha).
- **Inverse distance**: $w_i = \dfrac{1}{\text{distance}(\mathbf{x}, \mathbf{x}_i)}$.
- **Softmax weighting**: $w_i = \dfrac{\exp\big(-\text{distance}(\mathbf{x}, \mathbf{x}_i)\big)}{\sum_j \exp\big(-\text{distance}(\mathbf{x}, \mathbf{x}_j)\big)}$.

Q7.2 Show that L2-regularization can be obtained from a suitable prior by Bayesian inference (from the MAP estimate). [10]

Odpověď (česky):

Nastavení problému

- Mějme parametry modelu $\mathbf{w}$, na něž chceme položit *Gaussovský prior* $\mathbf{w} \sim \mathcal{N}(\mathbf{w}, \mathbf{0}, \sigma^2\mathbf{I})$.
- $p(\mathbf{w}) \;=\; \prod_i \mathcal{N}\big(w_i; 0, \sigma^2\big)$.

Maximum a posteriori (MAP)

$$\mathbf{w}_{\text{MAP}} = \arg\max_{\mathbf{w}} \Big[p(\mathbf{X} \mid \mathbf{w})\; p(\mathbf{w}) \Big] = \arg\min_{\mathbf{w}} \Big[-\log p(\mathbf{X} \mid \mathbf{w}) \;-\; \log p(\mathbf{w}) \Big].$$

- $-\log p(\mathbf{w})$ pro Gaussův prior $\sim \|\mathbf{w}\|^2$.

Výsledek

$$\mathbf{w}_{\text{MAP}} = \arg\min_{\mathbf{w}} \Big[\underbrace{-\sum_{i=1}^{N} \log p(\mathbf{x}_i \mid \mathbf{w})}_{\text{data-fitting}} \;+\; \underbrace{\frac{\|\mathbf{w}\|^2}{2\sigma^2}}_{\text{L2 regulace}} \Big].$$

Tím ukážeme, že L2-regulace ($\|\mathbf{w}\|^2$) odpovídá *Gaussovskému prioru* s nulovou střední hodnotou (tzv. ridge penalty).

Q7.3 Write down how $p(C_k \mid \mathbf{x})$ is approximated in a Naive Bayes classifier, explicitly state the Naive Bayes assumption, and show how the prediction is performed. [10]

Odpověď (česky):

Naivní Bayesova aproximace

- Předpoklad: x_d jsou *podmíněně nezávislé* vzhledem k třídě C_k. (Tzn. $p(\mathbf{x} \mid C_k) \approx \prod_d p(x_d \mid C_k)$.)

$$p(\mathbf{x} \mid C_k) \;=\; \prod_{d=1}^{D} p(x_d \mid C_k).$$

Bayesovo pravidlo

$$p(C_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_k)\; p(C_k)}{p(\mathbf{x})}, \quad \text{kde } p(\mathbf{x}) \text{ je konstanta pro všechny } k.$$

Klasifikační rozhodnutí

$$\hat{k} = \arg\max_k p(C_k \mid \mathbf{x}) = \arg\max_k p(\mathbf{x} \mid C_k)\, p(C_k).$$

- V praxi bereme logaritmus pro stabilitu: $\arg\max_k \log p(C_k) + \sum_{d=1}^{D} \log p(x_d \mid C_k)$.

Shrnutí: Naive Bayes je rychlý a výpočetně nenáročný; předpokládá, že rysy x_d jsou nezávislé v rámci jedné třídy C_k.

Q7.4 Considering a Gaussian naive Bayes, describe how $p(x_d \mid C_k)$ is modeled (what distribution and which parameters does it have) and how we estimate it during fitting. [10]

Odpověď (česky):

Gaussovský model rysů Pro Gaussovský Naive Bayes předpokládáme, že každá spojitá rysová hodnota x_d má normální rozdělení podmíněné třídou C_k. Pravděpodobnostní model má tvar:

$$p(x_d \mid C_k) = \mathcal{N}\Big(x_d;\, \mu_{d,k},\, \sigma^2_{d,k}\Big),$$

kde:

- $\mu_{d,k}$ je střední hodnota rysu x_d ve třídě C_k,
- $\sigma^2_{d,k}$ je rozptyl rysu x_d ve třídě C_k.

Odhad parametrů (Maximum Likelihood Estimation - MLE) Během trénování odhadujeme parametry $\mu_{d,k}$ a $\sigma^2_{d,k}$ pomocí maximální věrohodnosti (MLE) na základě trénovacích dat. Pro N_k vzorků ve třídě C_k platí:

$$\mu_{d,k} = \frac{1}{N_k} \sum_{i=1}^{N_k} x_{i,d},$$

$$\sigma^2_{d,k} = \frac{1}{N_k} \sum_{i=1}^{N_k} \big(x_{i,d} - \mu_{d,k}\big)^2.$$

Smoothing (hlazení) rozptylu V praxi se pro zajištění stability výpočtů a zabránění nulovému rozptylu často přidává smoothing term α, který mírně zvyšuje rozptyl:

$$\sigma^2_{d,k} = \frac{1}{N_k + \alpha} \sum_{i=1}^{N_k} \big(x_{i,d} - \mu_{d,k}\big)^2 + \alpha.$$

Například ve Scikit-learn je výchozí hodnota $\alpha = 10^{-9}$ násobek největšího rozptylu všech rysů.

Použití v Naive Bayes Pro klasifikaci celého vektoru $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ se pravděpodobnost podmíněná třídou C_k modeluje jako:

$$p(\mathbf{x} \mid C_k) = \prod_{d=1}^{D} \mathcal{N}(x_d; \mu_{d,k}, \sigma^2_{d,k}).$$

Predikce příslušnosti ke třídě C_k se pak určí maximalizací posteriorní pravděpodobnosti:

$$\hat{k} = \arg\max_k \Big[\log p(C_k) + \sum_{d=1}^{D} \log \mathcal{N}(x_d; \mu_{d,k}, \sigma^2_{d,k})\Big],$$

kde $p(C_k)$ je apriorní pravděpodobnost třídy C_k.

Shrnutí Gaussovský Naive Bayes je efektivní a jednoduchý klasifikační model, který předpokládá nezávislost rysů a normální rozdělení pro každou třídu. Klíčovým krokem je odhad parametrů $\mu_{d,k}$ a $\sigma^2_{d,k}$ během trénování a následné využití těchto parametrů pro výpočet pravděpodobností během klasifikace.

Q7.5 Considering a Bernoulli naive Bayes, describe how $p(x_d \mid C_k)$ is modeled (what distribution and which parameters does it have) and how we estimate it during fitting. [10]

Odpověď (česky):

Bernoulli model

- $x_d \in \{0, 1\}$ (binární rys).
- $p(x_d = 1 \mid C_k) = p_{d,k}, \quad p(x_d = 0 \mid C_k) = 1 - p_{d,k}$.

Pravděpodobnost třídy C_k daného x

$$p(\mathbf{x} \mid C_k) \;\propto\; \prod_{d=1}^{D} p_{d,k}^{x_d} \left(1 - p_{d,k}\right)^{1-x_d}$$

Logaritmus a predikce

$$\log p(C_k \mid \mathbf{x}) = \log p(C_k) + \sum_{d=1}^{D} \Big[x_d \log p_{d,k} + (1 - x_d) \log(1 - p_{d,k}) \Big].$$

$$\hat{k} = \arg\max_k \Big[b_k + \mathbf{x}^T \mathbf{w}_k \Big] \quad \text{(po sloučení konstant)}.$$

Odhad parametrů $p_{d,k}$

$$p_{d,k} = \frac{\sum_{i=1}^{N_k} x_{i,d}}{N_k}$$

- Opět se často přidává Laplace smoothing: $p_{d,k} = \dfrac{\sum_{i=1}^{N_k} x_{i,d} + \alpha}{N_k + 2\alpha}$.

Shrnutí: Bernoulli Naive Bayes se hodí pro binární rysy (přítomnost/nepřítomnost). Předpokládá nezávislost rysů, odhad pravděpodobností $p_{d,k}$ je prostý průměr plus smoothing pro zamezení nulových odhadů.

Q7.6 What measures can we take to prevent numeric instabilities in the Naive Bayes classifier, particularly if the probability density is too high in Gaussian Naive Bayes and there are zero probabilities in Bernoulli Naive Bayes? [10]

Odpověď (česky):

1. Problémy v Gaussian Naive Bayes (GNB):

- Pokud je pravděpodobnostní hustota příliš vysoká, mohou hodnoty $\mathcal{N}(x_d; \mu_{d,k}, \sigma^2_{d,k})$ způsobit numerické chyby (např. přetečení nebo podtečení).
- Řešení:
  - **Přechod na logaritmické pravděpodobnosti:** Místo násobení pravděpodobností sčítáme jejich logaritmy:

$$\log p(C_k \mid \mathbf{x}) = \log p(C_k) + \sum_{d=1}^{D} \log \mathcal{N}(x_d; \mu_{d,k}, \sigma^2_{d,k}).$$

  - **Přidání smoothing faktoru:** Upravíme odhad $\sigma^2_{d,k}$, aby nedošlo k nulovým nebo extrémně malým hodnotám:

$$\sigma^2_{d,k} = \frac{1}{N_k + \alpha} \sum_{i=1}^{N_k} (x_{i,d} - \mu_{d,k})^2 + \alpha.$$

2. Problémy v Bernoulli Naive Bayes (BNB):

- Zero probabilities ($p_{d,k} = 0$) způsobují, že celý součin pravděpodobností je nulový.
- Řešení:
  - **Laplace smoothing:** Přidáme k počtu výskytů hodnotu $\alpha > 0$, aby se zabránilo nulovým odhadům:

$$p_{d,k} = \frac{\sum_{i=1}^{N_k} x_{i,d} + \alpha}{N_k + 2\alpha}.$$

  - **Přechod na logaritmy:** Pro výpočet logaritmických pravděpodobností:

$$\log p(C_k \mid \mathbf{x}) = \log p(C_k) + \sum_{d=1}^{D} \big[x_d \log p_{d,k} + (1 - x_d) \log(1 - p_{d,k}) \big].$$

Shrnutí: Použití logaritmických pravděpodobností a smoothing technik výrazně snižuje numerické problémy a umožňuje stabilní výpočet Naive Bayes modelu.

Q7.7 What is the difference between discriminative and (classical) generative models? [5]

Odpověď (česky):

1. Discriminative models (diskriminační modely):

- **Co modelují:** Přímo se zaměřují na podmíněnou pravděpodobnost $p(C_k \mid \mathbf{x})$, tedy pravděpodobnost třídy C_k vzhledem k rysům $\mathbf{x}$.
- **Příklady:** Logistic Regression, Support Vector Machines (SVM), Neural Networks.
- **Vlastnosti:**
  - Obvykle poskytují vyšší přesnost predikce, protože přímo optimalizují rozhodovací hranice.
  - Nevyžadují modelování rozložení $\mathbf{x}$.

2. Generative models (generativní modely):

- **Co modelují:** Učí se společné rozložení $p(C_k, \mathbf{x})$, tj. pravděpodobnost rysů $\mathbf{x}$ a tříd C_k současně.
- **Příklady:** Naive Bayes, Hidden Markov Models (HMM), Gaussian Mixture Models (GMM).
- **Vlastnosti:**
  - Lze je použít pro generování nových dat $\mathbf{x}$.
  - Předpokládají nezávislost nebo jiné struktury mezi rysy, což může vést k nižší přesnosti při silné korelaci mezi rysy.

Shrnutí rozdílů:

- **Discriminative:** Optimalizují predikci třídy $\mathbf{x}$, přímý výpočet $p(C_k \mid \mathbf{x})$.
- **Generative:** Modelují společné rozložení $p(C_k, \mathbf{x})$ a odvozují $p(C_k \mid \mathbf{x})$ pomocí Bayesova pravidla.

Part VIII

Q8.1 Prove that independent discrete random variables are uncorrelated. [10]

Odpověď (česky):

Definice nezávislosti: Dvě diskrétní náhodné veličiny X a Y jsou nezávislé, pokud jejich společné rozdělení pravděpodobnosti lze vyjádřit jako součin marginálních rozdělení:

$$P(X = x, Y = y) = P(X = x) \cdot P(Y = y) \quad \forall x, y.$$

Definice kovariance: Kovariance náhodných veličin X a Y je definována jako:

$$\mathrm{Cov}(X, Y) = \mathbb{E}\big[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\big].$$

Důkaz: Pro nezávislé veličiny platí, že očekávání součinu je rovno součinu očekávání:

$$\mathbb{E}[XY] = \mathbb{E}[X] \cdot \mathbb{E}[Y].$$

Dosadíme do vzorce pro kovarianci:

$$\text{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X] \cdot \mathbb{E}[Y].$$

Protože $\mathbb{E}[XY] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$, dostáváme:

$$\text{Cov}(X, Y) = \mathbb{E}[X] \cdot \mathbb{E}[Y] - \mathbb{E}[X] \cdot \mathbb{E}[Y] = 0.$$

Závěr: Pokud jsou X a Y nezávislé, jejich kovariance je rovna nule, což znamená, že jsou nekorelované.

Q8.2 Write down the definition of covariance and Pearson correlation coefficient ρ, including its range. [10]

Kovariance: Kovariance mezi dvěma náhodnými proměnnými X a Y měří společnou variabilitu těchto proměnných. Je definována jako:

$$\text{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])],$$

kde $\mathbb{E}$ označuje očekávanou hodnotu.

Pearsonův korelační koeficient: Pearsonův korelační koeficient, označovaný jako ρ nebo r, je definován jako:

$$\rho = \frac{\text{Cov}(X, Y)}{\sqrt{\text{Var}(X)\text{Var}(Y)}},$$

$$r = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}},$$

kde:

- ρ se používá, pokud je vypočten s plným očekáváním (populační Pearsonův korelační koeficient),
- r se používá při odhadu koeficientu z dat (vzorkový Pearsonův korelační koeficient),
- $\bar{x}$ a $\bar{y}$ jsou vzorkové odhady středních hodnot.

Rozmezí: $\rho \in [-1, 1]$:

- Hodnota 1: Perfektní pozitivní lineární vztah.
- Hodnota -1: Perfektní negativní lineární vztah.
- Hodnota 0: Žádný lineární vztah.

Q8.3 Explain how Spearman's rank correlation coefficient and Kendall's rank correlation coefficient are computed. [10]

Spearmanův korelační koeficient: Spearmanův korelační koeficient ρ je neparametrická míra rankové korelace, která zkoumá statistickou závislost mezi pořadími dvou proměnných. Je definován jako:

$$\rho = 1 - \frac{6 \sum_i d_i^2}{n(n^2 - 1)},$$

kde:

- $d_i = \text{rank}(X_i) - \text{rank}(Y_i)$ je rozdíl pořadí odpovídajících hodnot,
- n je počet pozorování.

Kendallův korelační koeficient: Kendallův korelační koeficient τ měří ordinalní asociaci mezi dvěma proměnnými. Je definován jako:

$$\tau = \frac{|\{\text{pairs } i \neq j : x_j > x_i, y_j > y_i\}| - |\{\text{pairs } i \neq j : x_j > x_i, y_j < y_i\}|}{\binom{n}{2}},$$

kde:

- $\binom{n}{2}$ je počet všech dvojic (i, j), kde $i \neq j$,
- Čitatel vyjadřuje rozdíl mezi počtem konkordantních ($x_j > x_i, y_j > y_i$) a diskordantních ($x_j > x_i, y_j < y_i$) dvojic.

Rozmezí: Oba koeficienty mají hodnoty v intervalu $[-1, 1]$:

- Hodnota 1: Perfektní shoda.
- Hodnota -1: Perfektní nesoulad.
- Hodnota 0: Absence asociace.

Q8.4 Describe setups where a correlation coefficient might be a good evaluation metric. [5]

Příklady vhodného použití korelačního koeficientu:

1. **Shoda mezi anotátory (IAA):** Korelační koeficienty, jako Pearsonův nebo Spearmanův, mohou být použity k měření shody mezi hodnoceními různých anotátorů na spojité škále.
2. **Hodnocení prediktivního modelu:** V regresi korelační koeficient hodnotí, jak dobře předpovědi modelu odpovídají skutečným hodnotám.
3. **Výběr příznaků (Feature Selection):** Korelace identifikuje vztahy mezi příznaky v datasetu, což pomáhá odhalit redundantní informace.

4. **Analýza časových řad:** Korelační koeficienty analyzují vztahy mezi časovými řadami, například akciovými cenami nebo úrokovými sazbami.

5. **Psychometrie:** Korelace ověřuje spolehlivost psychometrických testů.

6. **Studie použitelnosti:** Korelační koeficienty hodnotí vztahy mezi metrikami použitelnosti, jako jsou časy na úkoly nebo míry chybovosti.

Poznámka: Korelační koeficienty indikují vztah mezi proměnnými, ale neimplikují kauzalitu. Měly by být používány spolu s dalšími metodami pro komplexnější hodnocení.

Q8.5 Describe under what circumstance correlation can be used to assess the validity of evaluation metrics. [5]

Odpověď (česky):

Korelace je zvláště užitečná pro ověřování platnosti evaluačních metrik v úlohách, kde chybí jasná ground truth. Lze ji aplikovat v následujících situacích:

- **SoTA (State-of-the-Art):** Benchmarking v soutěžích, kde metriky mají korelovat s odborným hodnocením pro přesné řazení účastníků.
- **Kontrola gramatiky:** Například hodnota β.
- **Validace nových metrik:** Validace proti zavedeným metrikám k zajištění, že měří podobné konstrukty.
- **Úkoly se subjektivním hodnocením:** Např. analýza sentimentu, kde metriky mají korelovat s lidskou intuicí.
- **Konzistence mezi datovými sadami:** Zajištění, že metrika je konzistentní napříč různými datovými sadami nebo podmínkami.

V těchto případech vysoká korelace s lidským úsudkem nebo mezi různými evaluačními metrikami naznačuje, že metriky zachycují aspekty výkonu odpovídající spolehlivému hodnocení.

Q8.6 Define Cohen's κ and explain what it is used for when preparing data for machine learning. [10]

Odpověď (česky):

Cohenův kappa koeficient (κ) je statistické měřítko pro hodnocení shody mezi hodnotiteli u kvalitativních (kategorických) položek. Definuje se jako:

$$\kappa = \frac{p_o - p_e}{1 - p_e},$$

kde p_o je relativní pozorovaná shoda mezi hodnotiteli a p_e je hypotetická pravděpodobnost shody náhodou.

Tento metr se používá v machine learningu k hodnocení konzistence anotací poskytovaných různými lidskými experty. Má následující účely:

- **Spolehlivost dat:** Zajišťuje, že datové labely používané pro trénování ML modelů jsou konzistentní a spolehlivé.
- **Výkon anotátorů:** Pomáhá při hodnocení výkonu anotátorů a umožňuje filtrovat nespolehlivé anotace.
- **Kultura a bias:** Nízké hodnoty κ mohou indikovat kulturní rozdíly nebo subjektivitu v datech.
- **Benchmarking modelů:** Nastavuje benchmark pro výkon ML modelů, protože vysoká přesnost nad rámec κ je často nereálná a může indikovat přeučení nebo únik dat.

Kvantifikací úrovně shody umožňuje Cohenovo κ informovanější rozhodování při přípravě dat, což vede k vývoji robustnějších modelů.

Q8.7 Considering an averaging ensemble of M models, prove the relation between the average mean squared error (MSE) of the ensemble and the average error of the individual models, assuming the model errors have zero means and are uncorrelated. [20]

Odpověď (česky):

Nechť $y_i(x)$ je predikce modelu i pro vstup x se skutečným cílem t, a nechť $\epsilon_i(x)$ je chyba modelu i, tj. $y_i(x) = t + \epsilon_i(x)$.

MSE jednotlivého modelu: Střední kvadratická chyba (MSE) modelu i je:

$$\mathbb{E}[(y_i(x) - t)^2] = \mathbb{E}[\epsilon_i(x)^2].$$

MSE ensemble: Pro ensemble M modelů je predikce:

$$y_{\text{ensemble}}(x) = \frac{1}{M} \sum_{i=1}^{M} y_i(x),$$

a jeho MSE je:

$$\mathbb{E}[(y_{\text{ensemble}}(x) - t)^2] = \mathbb{E}\left[\left(\frac{1}{M} \sum_{i=1}^{M} \epsilon_i(x) \right)^2 \right].$$

Za předpokladu nekorelovaných chyb: Pokud $\mathbb{E}[\epsilon_i(x)\epsilon_j(x)] = 0$ pro $i \neq j$ a chyby mají nulové střední hodnoty, platí:

$$\mathbb{E}\left[\left(\frac{1}{M} \sum_{i=1}^{M} \epsilon_i(x) \right)^2 \right] = \frac{1}{M^2} \sum_{i=1}^{M} \mathbb{E}[\epsilon_i(x)^2].$$

Proto je průměrná MSE ensemble:

$$\mathbb{E}[(y_{\text{ensemble}}(x) - t)^2] = \frac{1}{M} \mathbb{E}[\epsilon_i(x)^2].$$

Q8.8 Explain knowledge distillation: what it is used for, describe how it is done. [10]

Odpověď (česky):

Knowledge distillation je proces přenosu znalostí z velkého nebo komplexního modelu (učitele) na menší model (studenta). Je zvláště užitečný pro nasazování deep learning modelů na zařízeních s omezeným výpočetním výkonem.

Algoritmus:

1. Natrénujte velký model nebo ensemble modelů na datasetu, čímž vytvoříte **učitele**.
2. Použijte model učitele k vygenerování měkkých výstupů (soft target distributions) pro každou instanci datasetu.
3. Natrénujte studentský model tak, aby napodoboval distribuci výstupů učitele pomocí ztrátové funkce, která porovnává měkké cíle s predikcemi studenta.

Intuice: Distribuce pravděpodobností od učitele poskytuje bohatší signál než one-hot cíle, protože obsahuje informace o vztazích mezi třídami. Tento signál lze efektivně využít k trénování menšího modelu, který se lépe učí.

Ztrátová funkce: Ztrátová funkce kombinovaně hodnotí:

- **Cross-entropy loss** na tvrdých cílech (one-hot target).
- **Kullback-Leibler divergence** mezi měkkými cíli učitele a predikcemi studenta. Tyto měkké cíle jsou často vypočteny pomocí softmax funkce s vysokou teplotou T:

$$\text{softmax}_T(z_i) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}.$$

  Zvyšená teplota zdůrazňuje vztahy mezi třídami.

Rozdíl oproti standardnímu trénování:

- Standardní trénování používá tvrdé (one-hot) cíle, zatímco knowledge distillation využívá měkké cíle odvozené z učitele.
- Knowledge distillation zahrnuje dvě fáze: trénování učitele a následně trénování studenta, zatímco standardní trénování má pouze jednu fázi.
- U distillation se nastavuje teplota T v softmax funkci, což poskytuje jemnější pravděpodobnostní distribuci.

Q8.9 Assuming you have collected data for classification by letting people annotate data instances. How do you estimate a reasonable range for classifier performance? [5]

Odpověď (česky):

Při odhadu rozumného rozmezí výkonnosti klasifikátoru lze postupovat následovně:

- **Shoda mezi anotátory:** Nejprve spočítejte míru shody mezi jednotlivými anotátory (např. Cohenovo κ nebo F1 skóre). To poskytne horní hranici výkonnosti, které by mohl klasifikátor dosáhnout.
- **Náhodná přesnost:** Odhadněte dolní hranici výkonnosti pomocí náhodné klasifikace. Pro nevyvážené datové sady můžete vzít v úvahu baseline, jako je přesnost většinové třídy.
- **Lidská výkonnost:** Pokud jsou k dispozici data, porovnejte výkonnost klasifikátoru s průměrnou výkonností lidských anotátorů. To poskytne realistický benchmark.

Tímto způsobem lze odhadnout interval, ve kterém se očekává výkonnost klasifikátoru, a určit, zda je model výkonný ve vztahu k lidskému hodnocení a základní přesnosti.

Part IX

Q9.1 In a regression decision tree, state what values are kept in internal nodes, define the squared error criterion and describe how is a leaf split during training (without discussing splitting constraints). [10]

Odpověd (česky):

Vnitřní uzly: Ve stromu pro regresi vnitřní uzly reprezentují příslušnou vlastnost (feature) a prahovou hodnotu použitou pro rozdělení dat. Listy uchovávají předpověděnou hodnotu cílové proměnné, která je vypočtena jako průměr hodnot cíle u všech vzorků spadajících do daného listu.

Kriterium součtu kvadrátů: Kriterium pro chybu C_{SE} je definováno pro uzel T jako:

$$C_{SE}(T) = \sum_{i \in I_T} (t_i - \bar{t}_T)^2,$$

kde $\bar{t}_T$ je průměr cílových hodnot t_i vzorků v uzlu T a I_T je množina indexů vzorků patřících do uzlu T.

Rozdělení listu během trénování: Rozdělení listu probíhá následovně:

1. Projde se všechny vlastnosti a možné prahové hodnoty.
2. Rozdělí se uzel na dva potomky T_L a T_R podle vybrané vlastnosti a prahu.
3. Vypočte se snížení C_{SE} u dětí.
4. Zvolí se rozdělení s největším snížením C_{SE}.

Dodatek: V každém kroku pro aktuální uzel T se testují všechny možné kombinace featur a prahových hodnot, které by mohly uzel rozdělit. Pro každé možné rozdělení se spočítá snížení chyby C_{SE}, což je rozdíl mezi $C_{SE}(T)$ a součtem $C_{SE}(T_L) + C_{SE}(T_R)$. Poté se vybere to rozdělení, které toto snížení maximalizuje (tedy nejvíce zlepší predikci). Po provedení rozdělení se algoritmus rekurzivně aplikuje na nově vytvořené uzly T_L a T_R.

Proces pokračuje rekurzivně, dokud nejsou splněna kritéria zastavení, jako maximální hloubka stromu, minimální počet vzorků v listu nebo minimální zlepšení kritéria.

Q9.2 In a K-class classification decision tree, state what values are kept in internal nodes, define the Gini index and describe how is a node split during training (without discussing splitting constraints). [10]

Odpověd (česky):

Vnitřní uzly: V klasifikačním rozhodovacím stromu pro K třídy vnitřní uzly obsahují rozhodovací kritéria, tedy vlastnost a prahovou hodnotu, podle kterých se data rozdělí do dvou potomků.

Giniho index: Giniho index je měřítko používané k hodnocení čistoty uzlu. Pro uzel T je definován jako:

$$C_{\text{Gini}}(T) = |I_T| \sum_{k=1}^{K} p_T(k)(1 - p_T(k)),$$

kde $p_T(k)$ je podíl vzorků třídy k v uzlu T a I_T je množina indexů vzorků patřících do uzlu T.

Rozdělení uzlu během trénování: Rozdělení uzlu probíhá následovně:

1. Projdou se všechny vlastnosti a možné prahové hodnoty.
2. Vypočte se Giniho index pro děti vzniklé po rozdělení.
3. Vybere se vlastnost a prahová hodnota, které minimalizují kombinovaný Giniho index dětí.

Proces pokračuje rekurzivně, dokud nejsou splněna kritéria zastavení, jako maximální hloubka stromu nebo minimální velikost uzlu.

Q9.3 In a K-class classification decision tree, state what values are kept in internal nodes, define the entropy criterion and describe how is a node split during training (without discussing splitting constraints). [10]

Odpověd (česky):

Vnitřní uzly: Vnitřní uzly obsahují rozhodovací pravidla, obvykle vlastnost a prahovou hodnotu, která rozdělují dataset do podmnožin.

Entropie: Entropie je měřítko nečistoty v uzlu a je definována jako:

$$C_{\text{entropy}}(T) = -|I_T| \sum_{k=1}^{K} p_T(k) \log p_T(k),$$

kde $p_T(k)$ je podíl vzorků třídy k v uzlu T a I_T je množina indexů vzorků v uzlu T.

Rozdělení uzlu během trénování: Rozdělení uzlu probíhá následovně:

1. Pro každou vlastnost vypočítat potenciální rozdělení a odpovídající entropii.
2. Vybrat rozdělení, které způsobí největší snížení entropie (největší informační zisk).
3. Aplikovat tento proces rekurzivně na každý nový uzel, dokud nejsou splněna kritéria zastavení.

Q9.4 For binary classification, derive the Gini index from a squared error loss. [20]

Uvažujme binární klasifikaci s trénovacími příklady, které spadají do listového uzlu T. Označme $n_T(0)$ počet příkladů s cílovou hodnotou 0, $n_T(1)$ počet příkladů s cílovou hodnotou 1 a p_T jako podíl příkladů s cílovou hodnotou 1 v T, tj.

$$p_T = \frac{n_T(1)}{n_T(0) + n_T(1)}.$$

Kvadratická ztráta $L(p)$ pro predikci p je definována jako:

$$L(p) = \sum_{i \in I_T} (p - t_i)^2,$$

kde t_i je cílová hodnota pro i-tý příklad.

Minimalizací kvadratické ztráty (položíme derivaci rovnou nule) zjistíme, že optimální predikce p je průměrná cílová hodnota v T, tj. $p = p_T$. Ztráta pro tuto predikci je:

$$L(p_T) = \sum_{i \in I_T} (p_T - t_i)^2 = n_T(0)(p_T - 0)^2 + n_T(1)(p_T - 1)^2.$$

Rozšířením výrazů dostáváme:

$$L(p_T) = n_T(0)p_T^2 + n_T(1)(1 - p_T)^2 = (n_T(0) + n_T(1))p_T(1 - p_T) = |I_T|p_T(1 - p_T).$$

Z toho plyne, že $G(T) = 2p_T(1 - p_T)$ je úměrný Giniho indexu pro binární klasifikaci.

Dodatečné vysvětlení: Při optimalizaci kvadratické ztráty používáme hodnotu p jako obecnou proměnnou, kterou nastavujeme tak, aby minimalizovala ztrátu. Výsledná optimalizovaná hodnota p_T odpovídá empirické pravděpodobnosti cílové hodnoty 1 v uzlu T. Tím pádem p_T reflektuje skutečné rozložení cílových hodnot v uzlu T, což přímo přispívá k výpočtu Giniho indexu.

Q9.5 For K-class classification, derive the entropy criterion from a non-averaged NLL loss. [20]

Mějme množinu trénovacích příkladů I_T, které odpovídají listovému uzlu T v rozhodovacím stromu pro K-třídní klasifikaci. Označme $n_T(k)$ jako počet příkladů v T s cílovou třídou k. Pravděpodobnost třídy k v T je definována jako:

$$p_T(k) = \frac{n_T(k)}{|I_T|}.$$

Negativní logaritmická ztráta (NLL) pro distribuci p nad K třídami je definována jako:

$$L(p) = \sum_{i \in I_T} -\log p_{t_i},$$

kde p_{t_i} je predikovaná pravděpodobnost pro skutečnou třídu t_i i-tého příkladu.

Minimalizací NLL ztráty získáme podmínku $p_k = p_T(k)$. Hodnota ztráty s ohledem na p_T je pak:

$$L(p_T) = \sum_{i \in I_T} -\log p_{t_i} = -\sum_{k:p_T(k)\neq 0} n_T(k) \log p_T(k).$$

Pomocí definice entropie $H(p_T)$ pro distribuci p_T máme:

$$H(p_T) = -\sum_{k:p_T(k)\neq 0} p_T(k) \log p_T(k),$$

což implikuje, že NLL ztráta je rovna velikosti I_T násobené entropií p_T:

$$L(p_T) = |I_T| \cdot H(p_T).$$

Tím je ukázáno, že minimalizace ne-průměrované NLL ztráty odpovídá minimalizaci entropie predikované distribuce tříd v listovém uzlu rozhodovacího stromu.

Q9.6 Describe how is a random forest trained (including bagging and a random subset of features) and how is prediction performed for regression and classification. [10]

Random forest je metoda učení ansámblů, která funguje na principu konstrukce velkého počtu rozhodovacích stromů během trénování a výstupu třídy, která je nejčastěji volena ze všech tříd (klasifikace), nebo průměru predikcí (regrese) jednotlivých stromů.

Trénování Proces trénování zahrnuje následující kroky:

1. **Bootstrap Aggregating (Bagging):** Pro každý strom je vytvořen bootstrapový vzorek z trénovacích dat. To znamená, že pro trénovací množinu velikosti N je M vzorků vybráno s náhradou pro vytvoření trénovací množiny stromu.

2. **Náhodný výběr featur:** Při dělení uzlů během konstrukce stromů se místo hledání nejlepšího dělení mezi všemi featurami zvolí náhodná podmnožina featur. Nejlepší dělení je pak nalezeno v rámci této podmnožiny, což zvyšuje diverzitu mezi stromy a přispívá k úspěchu náhodných lesů.

3. **Konstrukce stromu:** Rozhodovací stromy jsou konstruovány do maximální hloubky bez prořezávání. Každý strom roste na odlišném bootstrapovém vzorku dat a na každém uzlu je pro dělení zvažována jiná náhodná podmnožina featur.

4. **Tvorba ansámblu:** Kroky 1 až 3 se opakují pro vytvoření lesu rozhodovacích stromů, obvykle v řádu desítek až stovek stromů.

Predikce Pro predikci jsou odpovědi všech stromů v lese agregovány:

- **Regrese:** Konečná predikce je průměr predikcí ze všech jednotlivých stromů.
- **Klasifikace:** Každý strom hlasuje pro třídu a třída s většinou hlasů se stává predikcí modelu. V případě nerozhodnosti může být třída náhodně vybrána, nebo může být rozhodnutí rozděleno na základě distribuce tříd.

Random forest využívá síly více rozhodovacích stromů ke snížení variance a vyhnutí se přeučení, poskytuje robustní predikce pro úlohy klasifikace i regrese.

Q9.7 Explain the CART algorithm for constructing a decision tree. Explain the relationship between the loss function that is optimized during the decision tree construction and the splitting criterion that is used during the node splitting. [10]

Popis CART algoritmu: CART (Classification and Regression Trees) je algoritmus používaný pro konstrukci rozhodovacích stromů, který lze aplikovat jak na úlohy klasifikace, tak regrese. Proces zahrnuje následující kroky:

1. **Výběr splitting kritéria:** Pro každý uzel algoritmus iteruje přes všechny featury a možné prahové hodnoty (thresholds), aby našel nejlepší možné dělení dat. V případě klasifikace se jako splitting kritérium používá Giniho index nebo entropie. U regrese se optimalizuje kvadratická chyba (squared error).

2. **Rozdělení uzlu:** Data jsou rozdělena na dvě podmnožiny na základě vybrané featury a prahové hodnoty, která maximalizuje pokles hodnoty splitting kritéria.

3. **Rekurzivní aplikace:** Proces se opakuje rekurzivně na každé podmnožině, dokud není dosaženo stopping kritéria, například maximální hloubky stromu, minimální velikosti uzlu nebo minimální hodnoty poklesu splitting kritéria.

4. **Predikce na listech:** U listů rozhodovacího stromu se v případě regrese predikovaná hodnota počítá jako průměr hodnot cílové proměnné v listu. V případě klasifikace se určuje nejpravděpodobnější třída na základě distribuce tříd v listu.

Vztah mezi optimalizovanou ztrátovou funkcí a splitting kritériem: Optimalizovaná ztrátová funkce při konstrukci rozhodovacího stromu přímo souvisí se splitting kritériem. V každém kroku algoritmus vyhledává dělení, které minimalizuje hodnotu ztrátové funkce na základě splitting kritéria:

- **Klasifikace:** Pro klasifikaci se nejčastěji používá Giniho index nebo entropie jako splitting kritéria. Obě metriky měří nečistotu (impurity) v uzlu. Minimalizací těchto metrik se redukuje chyba klasifikace na trénovacích datech.

- **Regrese:** V regresi se jako splitting kritérium používá kvadratická chyba (squared error). Minimalizace této chyby zajišťuje, že průměr cílových hodnot v listech co nejlépe odpovídá skutečným hodnotám.

Rozdíl mezi splitting kritériem a celkovou ztrátovou funkcí je v jejich aplikačním měřítku. Splitting kritérium se používá lokálně na úrovni jednotlivých uzlů, zatímco ztrátová funkce hodnotí celý strom jako celek. Přesto je cílem splitting kritéria jeho optimalizace tak, aby přispěla k minimalizaci globální ztrátové funkce.

Part X

Q10.1 Write down the loss function which we optimize in gradient boosted decision trees during the construction of *t*-th tree. Then define g_i and h_i and show the value w_T of optimal prediction in node T and the criterion used during node splitting. [20]

Odpověď: Při konstrukci $t$ého stromu v gradient boosting optimalizujeme následující ztrátovou funkci:

$$E^{(t)}(w_t; w_{1..t-1}) = \sum_i \Big[\ell(t_i, y^{(t-1)}(x_i; w_{1..t-1}) + y_t(x_i; w_t))\Big] + \frac{1}{2}\lambda||w_t||^2,$$

kde přibližně platí:

$$E^{(t)}(w_t; w_{1..t-1}) \approx \sum_i \Big[\ell(t_i, y^{(t-1)}(x_i)) + g_i \cdot y_t(x_i) + \frac{1}{2}h_i \cdot y_t^2(x_i)\Big] + \frac{\lambda}{2}||w||^2.$$

g_i a h_i jsou definovány jako první a druhé parciální derivace ztrátové funkce vzhledem k predikci v kroku $t-1$, konkrétně:

$$g_i = \frac{\partial \ell(t_i, y)}{\partial y}\Big|_{y=y^{(t-1)}(x_i)},$$

$$h_i = \frac{\partial^2 \ell(t_i, y)}{\partial y^2}\Big|_{y=y^{(t-1)}(x_i)}.$$

Optimální predikce w_T pro uzel T je dána vztahem:

$$w_T^* = -\frac{\sum_{i\in I_T} g_i}{\lambda + \sum_{i\in I_T} h_i},$$

kde I_T označuje množinu indexů příkladů v uzlu T. Během dělení uzlu maximalizujeme přínos rozdělení ("gain"), který je vypočítán s využitím těchto gradientů.

Kritérium pro dělení uzlu: Kritérium pro dělení uzlu v gradient boosting je založeno na maximalizaci přínosu G, který je vypočítán následovně:

$$G = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\lambda + \sum_{i\in I_L} h_i} + \frac{(\sum_{i\in I_R} g_i)^2}{\lambda + \sum_{i\in I_R} h_i} - \frac{(\sum_{i\in I_T} g_i)^2}{\lambda + \sum_{i\in I_T} h_i}\right] - \gamma,$$

kde I_L a I_R jsou množiny indexů instancí pro levý a pravý split z uzlu T, a γ je regulační parametr penalizující složitost modelu.

Q10.2 For a K-class classification, describe how to perform prediction with a gradient boosted decision tree trained for T time steps (how the individual trees perform prediction and how are the KT trees combined to produce the predicted categorical distribution). [10]

Odpověď: Pro klasifikační úlohu s K třídami, kde je použit gradient boosting s rozhodovacími stromy, se predikce po T časových krocích provádí následovně:

- Každý časový krok t zahrnuje trénování K stromů, označených jako $w_{t,k}$, přičemž každý strom predikuje hodnotu pro jednu z K tříd.
- Predikce pro instanci x_i je získána součtem výstupů všech stromů odpovídajících každé třídě přes všechny časové kroky:

$$\text{softmax}(y(x_i)) = \text{softmax}\Big(\sum_{t=1}^{T}\big[y_{t,1}(x_i; w_{t,1}), \ldots, y_{t,K}(x_i; w_{t,K})\big]\Big),$$

  kde $y_{t,k}(x_i; w_{t,k})$ je výstup k-tého stromu v časovém kroku t pro instanci x_i.

Predikovaná kategorická distribuce pro instanci x_i je tedy výstup softmax funkce, která poskytuje pravděpodobnostní rozdělení přes K třídy.

Q10.3 What type of data are gradient boosted decision trees good for as opposed to multilayer perceptron? Explain the intuition why it is the case. [5]

Odpověď: Gradient boosted decision trees (GBDTs) jsou optimální pro strukturovaná, tabulková data, kde mají vstupní featury vysokou prediktivní sílu a jasnou interpretovatelnost. Mohou zachytit nelineární vztahy a interakce mezi featurami bez nutnosti rozsáhlého předzpracování.

Výhody GBDTs:

- Vhodné pro data s nízkou dimenzionalitou s významnými featurami.
- Pracují s různými typy dat (kontinuální, kategoriální).
- Vyžadují méně předzpracování dat.

Na druhou stranu, vícevrstvé perceptrony (MLPs), neboli neuronové sítě, jsou vhodnější pro data s vysokou dimenzionalitou, jako jsou obrázky nebo texty. Tyto modely excelují v učení hierarchických reprezentací featur, což je zásadní v doménách, kde jednotlivé featury samy o sobě nejsou informativní.

Výhody MLPs:

- Vhodné, když jednotlivé featury samy o sobě nemají velký význam.
- Ideální pro data s vysokou dimenzionalitou (obrázky, texty).
- Schopné extrahovat složité featury.
- Možnost využití předtrénované sítě.

Výběr modelu závisí na charakteristikách datasetu a daném problému.

Q10.4 Explain the main differences between random forests and gradient-boosted decision trees. [5]

Odpověď: Hlavní rozdíly mezi random forests (RF) a gradient-boosted decision trees (GBDT) jsou následující:

- **Metoda trénování:**
  - RF trénuje mnoho nezávislých stromů paralelně na bootstrapovaných vzorcích dat a kombinuje jejich predikce (agregací pro regresi nebo hlasováním pro klasifikaci).
  - GBDT trénuje stromy sekvenčně, kde každý nový strom koriguje chyby předchozího modelu.
- **Redukce variance vs. bias:**
  - RF se zaměřuje na redukci variance (kombinací více stromů a rozdělováním tréninkových vzorků).
  - GBDT se zaměřuje na redukci biasu (postupnou optimalizací ztrátové funkce).
- **Rychlost trénování:**
  - RF je obvykle rychlejší na trénování, protože stromy jsou trénovány paralelně.
  - GBDT je pomalejší, protože vyžaduje sekvenční trénování stromů.
- **Robustnost:**
  - RF je robustní vůči šumu v datech, protože využívá bootstrap a agregaci.
  - GBDT může být citlivější na šum v datech, pokud není správně regularizován.
- **Aplikace:**
  - RF je vhodný pro rychlé predikce a problémy s vysokou variancí.
  - GBDT je vhodný pro problémy s vysokým biasem a pro úlohy, kde je důležitá vysoká přesnost.

Q10.5 Explain the intuition for second-order optimization using Newton's root-finding method or Taylor expansions. [10]

Odpověď: Druhořadá optimalizace využívá druhou derivaci ztrátové funkce ke zlepšení rychlosti a přesnosti konvergence. Zároveň se pojmenování a princip opírají o Taylorovu aproximaci 2. řádu. Princip funguje následujícím způsobem:

- **Newtonova metoda hledání kořenů:**
  - Newtonova metoda iterativně hledá kořeny derivace funkce $f'(x)$ pomocí gradientu ($f'(x)$) a druhé derivace ($f''(x)$).
  - Aktualizační pravidlo je:
$$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}.$$
  - Intuice této metody spočívá v tom, že druhá derivace ($f''(x)$) koriguje krok na základě zakřivení funkce, což umožňuje rychlejší sbližování s kořenem (nebo extrémem, pokud je metoda aplikována na derivaci ztrátové funkce $\ell'(x)$).
- **Taylorova expanze:**
  - Taylorova expanze druhého řádu aproximuje hodnotu funkce $f(x)$ v okolí bodu x_0:
$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2.$$
  - Druhá derivace $f''(x_0)$ přidává informaci o zakřivení funkce, což umožňuje lépe odhadnout, jak velký krok je vhodné udělat.
  - V kontextu optimalizace je cílem minimalizovat $f(x)$, což odpovídá nalezení bodu, kde derivace $f'(x)$ je rovna nule. Taylorova expanze pomáhá tento bod odhadnout.
- **Aplikace v optimalizaci:**
  - **Optimalizace prvního řádu:** Gradient sestupu (SGD) využívá pouze první derivaci $g_i = \frac{\partial \ell}{\partial y}$, aby určil směr minimalizace ztrátové funkce.
  - **Optimalizace druhého řádu:** Druhořadý gradient ($h_i = \frac{\partial^2 \ell}{\partial y^2}$) přidává informaci o zakřivení funkce. To umožňuje nejen určit směr minimalizace, ale i efektivní velikost kroku (adaptive step size).
  - V gradient-boosted decision trees (GBDT) se tyto derivace využívají k aktualizaci váhových parametrů stromů. Nové stromy opravují chyby na základě gradientu a zakřivení (hessianu) předchozích stromů.

Shrnutí: Optimalizace druhého řádu je rychlejší a přesnější než optimalizace prvního řádu (SGD), protože zohledňuje zakřivení ztrátové funkce. Tato metoda však vyžaduje výpočet druhé derivace, což zvyšuje výpočetní náročnost. Výhodou je rychlejší konvergence k optimálnímu řešení, což je klíčové pro algoritmy jako GBDT nebo při řešení složitých nelineárních problémů.

Part XI

Q11.1 Formulate SVD decomposition of matrix X, describe properties of individual parts of the decomposition. Explain what the reduced version of SVD is. [10]

Odpověd: SVD (Singular Value Decomposition) matice $X \in \mathbb{R}^{m \times n}$ je rozklad ve tvaru:

$$X = U\Sigma V^T,$$

kde:

- $U \in \mathbb{R}^{m \times m}$ je ortogonální matice, jejíž sloupce jsou levé singulární vektory matice X,
- $\Sigma \in \mathbb{R}^{m \times n}$ je diagonální matice s nezápornými singulárními hodnotami seřazenými sestupně,
- $V \in \mathbb{R}^{n \times n}$ je ortogonální matice, jejíž sloupce jsou pravé singulární vektory matice X.

Vlastnosti:

- Sloupce U a V tvoří ortonormální báze sloupcového a řádkového prostoru matice X.
- Nenulové singulární hodnoty v Σ odpovídají odmocninám nenulových vlastních čísel matic $X^T X$ a XX^T.

Interpretace bází:

- **Báze vektorového prostoru:** Báze je minimální sada vektorů, která popisuje celý prostor. Vektory v bázi jsou lineárně nezávislé (nejdou odvodit jeden od druhého) a mohou, ale nemusí být na sebe kolmé.
- U: Tvoří ortonormální bázi sloupcového prostoru matice X, tedy prostoru definovaného všemi lineárními kombinacemi sloupců matice X.
- V: Tvoří ortonormální bázi řádkového prostoru matice X, tedy prostoru definovaného všemi lineárními kombinacemi řádků matice X.

Redukovaná SVD: Redukovaná verze SVD se používá, pokud chceme aproximovat matici X maticí nižší hodnosti (nižšího ranku) k, kde $k < \min(m, n)$:

$$\tilde{X} = U_k \Sigma_k V_k^T,$$

kde U_k a V_k obsahují pouze první k sloupců z U a V a Σ_k obsahuje pouze prvních k singulárních hodnot. Tento proces odstraní směry, které přispívají nejméně k informacím v matici X, a zachová nejdůležitější strukturu dat.

—

Q11.2 Formulate the Eckart-Young theorem. Provide an interpretation of what the theorem says and why it is useful. [10]

Odpověď: Eckart-Youngův teorém uvádí, že pro danou matici $X \in \mathbb{R}^{m \times n}$ a její dekompozici SVD je nejlepší aproximace matice X_k o hodnosti k z hlediska Frobeniovy normy dána vztahem:

$$X_k = \sigma_1 u_1 v_1^T + \cdots + \sigma_k u_k v_k^T,$$

kde σ_i jsou singulární hodnoty a u_i, v_i jsou odpovídající levé a pravé singulární vektory.

Význam: Tento teorém minimalizuje Frobeniovu normu rozdílu mezi X a X_k:

$$\|X - X_k\|_F \leq \|X - B\|_F,$$

pro jakoukoliv matici B hodnosti k. Je užitečný v úlohách komprese dat a redukce dimenzionality.

—

Q11.3 Explain how to compute the PCA of dimension M using the SVD decomposition of a data matrix X, and why it works. [10]

Odpověď: Hlavní komponenty PCA lze získat pomocí SVD matice X:

1. Data zcentrujeme odečtením průměru každého atributu: $\tilde{X} = X - \mu$.
2. Provedeme SVD na matici $\tilde{X}$: $\tilde{X} = U\Sigma V^T$.
3. Sloupce matice V odpovídají hlavním komponentám dat.
4. Vybereme prvních M sloupců V a prvních M singulárních hodnot z Σ.
5. Provedeme projekci dat: $X_{\text{PCA}} = XV_M$, kde V_M obsahuje prvních M sloupců V.

Proč to funguje: Singulární hodnoty v Σ reprezentují množství variance zachycené každou hlavní komponentou. Projekcí na M hlavních komponent zachováváme maximální množství variance v datech.

—

Q11.4 Given a data matrix X, write down the algorithm for computing the PCA of dimension M using the power iteration algorithm. [20]

Odpověď: Algoritmus:

1. Data zcentrujeme: $\tilde{X} = X - \mu$.
2. Vypočítáme kovarianční matici: $S = \frac{1}{N}(\tilde{X})^T \tilde{X}$.
3. Pro každou dimenzi i od 1 do M:
   (a) Inicializujeme náhodný vektor v_i.

(b) Iterativně hledáme vlastní vektor:

$$v_i = Sv_i,\ \lambda_i = \|v_i\|,\ v_i = \frac{v_i}{\lambda_i}.$$

(c) Deflace: $S = S - \lambda_i v_i v_i^T$.

4. Matice V obsahuje $v_1, v_2, \ldots, v_M$ jako hlavní komponenty.

5. Provedeme projekci: $X_{\text{PCA}} = XV$.

—

Q11.5 List at least two applications of SVD or PCA. [5]

Odpověď:

1. Redukce dimenzionality: PCA se používá k odstranění šumu a snížení počtu atributů v datové sadě.

2. Doplnění chybějících dat: SVD pomáhá rekonstruovat chybějící hodnoty v maticích.

—

Q11.6 Describe the K-means algorithm, including the kmeans++ initialization. What is it used for? What is the loss function that the algorithm optimizes? What can you say about the algorithm convergence? [20]

Odpověď: K-means je shlukovací algoritmus, který slouží k rozdělení datových bodů do K shluků (clusterů) tak, aby se minimalizoval vnitroshlukový součet čtverců vzdáleností (intra-cluster sum of squared distances). Cílem algoritmu je optimalizovat následující ztrátovou funkci:

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{i,k} \|x_i - \mu_k\|^2,$$

kde:

- x_i jsou vstupní data (vektory v prostoru $\mathbb{R}^D$),
- μ_k jsou středy shluků (centra clusterů),
- $z_{i,k} \in \{0, 1\}$ je binární proměnná, která určuje, zda bod x_i náleží do shluku k ($z_{i,k} = 1$, pokud bod patří do clusteru k, jinak $z_{i,k} = 0$).

Postup algoritmu K-means:

1. **Inicializace:** Nejprve náhodně vybereme K počátečních center shluků $\mu_1, \mu_2, \ldots, \mu_K$. Alternativně lze použít **kmeans++ inicializaci**:

   - První centrum se vybere náhodně z datových bodů.

- Každé další centrum se vybírá s pravděpodobností úměrnou vzdálenosti bodu k nejbližšímu již zvolenému centru (čím dále je bod, tím větší šance, že bude zvolen).
- Výsledkem je lepší počáteční rozmístění center a rychlejší konvergence algoritmu.

2. **Přiřazení bodů:** Každý bod x_i je přiřazen k nejbližšímu centru μ_k podle euklidovské vzdálenosti:
$$z_{i,k} = \begin{cases} 1 & \text{pro } k = \arg\min_j \|x_i - \mu_j\|^2, \\ 0 & \text{jinak.} \end{cases}$$

3. **Aktualizace center:** Nová poloha center shluků je vypočtena jako průměr bodů přiřazených ke každému centru:
$$\mu_k = \frac{\sum_{i=1}^{N} z_{i,k} x_i}{\sum_{i=1}^{N} z_{i,k}}.$$

4. **Opakování:** Kroky přiřazení a aktualizace se opakují, dokud nedojde ke konvergenci (tj. centra shluků se přestanou měnit nebo se hodnota ztrátové funkce J již dále nesnižuje).

Konvergence:

K-means algoritmus vždy konverguje, protože každý krok (přiřazení bodů a aktualizace center) snižuje hodnotu ztrátové funkce J. Nicméně algoritmus může uvíznout v lokálním minimu, což závisí na počátečním rozmístění center shluků. Použití $kmeans++$ inicializace zvyšuje pravděpodobnost nalezení globálně lepšího řešení.

Využití:

Algoritmus K-means se využívá v oblastech jako:

- analýza zákazníků (např. segmentace trhu),
- předzpracování dat (např. komprese obrazu),
- vyhledávání vzorů ve velkých datových sadách.

—

Q11.7 Name at least two clustering algorithms. What is their main principle? How do they differ? [10]

Odpověď:

1. K-means: Rozděluje data do K shluků minimalizací vnitroshlukového součtu čtverců.
2. Hierarchické shlukování: Postupně spojuje (aglomerativní) nebo rozděluje (divizivní) shluky na základě vzdáleností.

Rozdíly: K-means vyžaduje předem daný počet shluků, zatímco hierarchické shlukování vytváří dendrogram, který umožňuje flexibilní volbu počtu shluků.

Part XII

Q12.1 Considering statistical hypothesis testing, define type I errors and type II errors (in terms of the null hypothesis). Finally, define what a significance level is. [10]

Type I error nastává, když zamítneme nulovou hypotézu (H_0), přestože je pravdivá. Je to tzv. "falešně pozitivní"chyba.

Type II error nastává, když nezamítneme nulovou hypotézu (H_0), přestože je nepravdivá. Je to tzv. "falešně negativní"chyba.

Significance level (hladina významnosti) je předem stanovená pravděpodobnost, se kterou jsme ochotni akceptovat Type I error (maximální míra rizika). Typicky se označuje jako α (např. 0,05) a určuje hranici p-hodnoty, pod kterou zamítáme nulovou hypotézu.

Q12.2 Explain what a test statistic and a p-value are. [10]

- Test statistic (testovací statistika) je hodnota vypočítaná z dat, která slouží k rozhodování o nulové hypotéze. Odráží odchylku pozorovaných dat od předpokladů nulové hypotézy.
- P-value (p-hodnota) je pravděpodobnost, že bychom pozorovali stejnou nebo extrémnější hodnotu testovací statistiky, pokud by nulová hypotéza byla pravdivá. Čím nižší p-hodnota, tím silnější důkaz proti nulové hypotéze.

Q12.3 Write down the steps of a statistical hypothesis test, including a definition of a p-value. [10]

1. Formulace hypotéz:
   - Nulová hypotéza (H_0): předpoklad, že žádný efekt neexistuje.
   - Alternativní hypotéza (H_1): předpoklad, že efekt existuje.
2. Výběr hladiny významnosti α (např. 0,05).
3. Výpočet testovací statistiky na základě dat.
4. Určení p-hodnoty:
   - P-hodnota je pravděpodobnost pozorování stejné nebo extrémnější hodnoty testovací statistiky za předpokladu platnosti H_0.
5. Rozhodnutí:
   - Pokud p-value $< \alpha$, zamítáme H_0.
   - Jinak H_0 nezamítáme.

Q12.4 Explain the differences between a one-sample test, a two-sample test, and a paired test. [10]

- One-sample test: Testuje, zda průměr jedné skupiny dat odpovídá konkrétní hodnotě (např. průměr = 10).
- Two-sample test: Porovnává průměry dvou nezávislých skupin dat (např. muži vs. ženy).
- Paired test: Porovnává průměry dvou závislých skupin dat, kde data z jedné skupiny odpovídají datům z druhé (např. měření před a po léčbě u stejných subjektů).

Q12.5 When considering the multiple comparison problem, define the family-wise error rate and prove the Bonferroni correction, which allows limiting the family-wise error rate by a given α. [10]

Family-wise error rate (FWER) je pravděpodobnost, že při provádění více statistických testů uděláme alespoň jednu chybu typu I (falešně pozitivní). FWER lze formálně zapsat jako:

$$\text{FWER} = P\left(\bigcup_i (p_i \leq \alpha)\right),$$

kde p_i jsou p-hodnoty jednotlivých testů.

Jednou z metod kontroly FWER je Bonferroniho korekce, která upravuje hladinu významnosti jednotlivých testů tak, aby celková FWER nepřekročila danou hodnotu α. Pro m testů je hladina významnosti každého testu nastavena na:

$$\alpha' = \frac{\alpha}{m}.$$

Použitím Booleovy nerovnosti lze dokázat, že Bonferroniho korekce zajišťuje kontrolu nad FWER:

$$P\left(\bigcup_i A_i\right) \leq \sum_i P(A_i),$$

kde A_i je událost, že $p_i \leq \alpha'$. Proto platí:

$$\text{FWER} = P\left(\bigcup_i (p_i \leq \alpha')\right) \leq \sum_i P(p_i \leq \alpha') = m \cdot \alpha' = m \cdot \frac{\alpha}{m} = \alpha.$$

Bonferroniho korekce je jednoduchá a konzervativní metoda, která zajišťuje, že celková pravděpodobnost chyby typu I nepřekročí požadovanou hladinu významnosti α. Existují však i jiné metody, jako Holm-Bonferroni nebo Šidákova korekce, které jsou méně konzervativní.

Q12.6 For a trained model and a given test set with N examples and metric E, write how to estimate 95% confidence intervals using bootstrap resampling. [10]

Bootstrap resampling je metoda, která nám umožňuje odhadnout spolehlivost modelu tím, že generujeme různé testovací sady opakovaným náhodným výběrem (s opakováním) z původní testovací sady.

1. Proveď bootstrap resampling:
   - Vytvoř R bootstrap vzorků testovací sady výběrem N příkladů s opakováním.
2. Pro každý vzorek:
   - Spočítej metrickou hodnotu E (např. přesnost modelu).
3. Vytvoř rozložení hodnot E z R vzorků.
4. Urči 95% interval spolehlivosti:
   - Seřaď hodnoty E a vezmi 2,5. percentil a 97,5. percentil rozložení jako hranice intervalu spolehlivosti.
5. Interpretace intervalu:
   - Pokud intervaly spolehlivosti dvou modelů (např. modelu A a modelu B) **nepřekrývají**, znamená to, že je statisticky významný rozdíl mezi jejich výkony.
   - Pokud se intervaly **překrývají**, nemůžeme rozhodnout, zda je jeden model lepší, protože rozdíl může být způsoben náhodou.

Q12.7 For two trained models and a given test set with N examples and metric E, explain how to perform a paired bootstrap test that the first model is better than the other. [10]

Paired bootstrap test je metoda, která přímo porovnává rozdíly ve výkonech dvou modelů na stejných testovacích vzorcích, což eliminuje vliv variability způsobené obtížností jednotlivých vzorků.

1. Vytvoř R bootstrap vzorků testovací sady (N příkladů s opakováním).
2. Pro každý vzorek:
   - Spočítej metrické hodnoty E pro oba modely.
   - Spočítej rozdíl výkonů $E(A) - E(B)$.
3. Vytvoř rozložení rozdílů $E(A) - E(B)$ z R vzorků.
4. Vyhodnocení:
   - Spočítej podíl rozdílů ≤ 0 (tj. model B je lepší nebo stejně dobrý).
   - Pokud je tento podíl $< 0,05$, zamítáme H_0 a usuzujeme, že model A je lepší.

Q12.8 For two trained models and a given test set with N examples and metric E, explain how to perform a random permutation test that the first model is better than the other with a significance level α. [10]

Permutation test je metoda, která testuje hypotézu, že rozdíly mezi dvěma modely jsou způsobeny náhodou. Tato metoda zahrnuje náhodné přehazování předpovědí mezi modely a měření rozdílů jejich výkonů.

1. Spočítej původní rozdíl výkonů $E(A) - E(B)$ na celé testovací sadě.
2. Proveď R permutací:
   - Náhodně prohoď předpovědi mezi modely (např. některé předpovědi modelu A zaměníš s předpověďmi modelu B).
   - Spočítej rozdíl výkonů $E(A) - E(B)$ pro každou permutaci.
3. Vytvoř rozložení rozdílů z permutací.
4. P-hodnota:
   - Spočítej podíl permutací, kde rozdíl $\geq$ původní rozdíl.
   - Pokud p-hodnota $< \alpha$, zamítáme H_0 a usuzujeme, že model A je lepší.

Part XIII

Q13.1: Explain the difference between deontological and utilitarian ethics. List examples of how these theoretical frameworks can be applied in machine learning ethics. [10]

Deontologická etika se zaměřuje na správnost nebo nesprávnost jednotlivých činů na základě pravidel nebo povinností. Například v kontextu strojového učení může být deontologickým přístupem zajistit, že algoritmus nikdy nebude diskriminovat určitou skupinu lidí, bez ohledu na výsledné důsledky.

Naopak utilitaristická etika se soustředí na maximalizaci celkového dobra nebo užitku, bez ohledu na prostředky, jakými bylo tohoto výsledku dosaženo. V aplikacích strojového učení by například utilitaristický přístup mohl ospravedlnit určité diskriminační výsledky modelu, pokud by vedly k většímu celkovému prospěchu společnosti.

Příklady:

- Deontologický přístup: Vytvoření modelu, který explicitně kontroluje, aby nebyl zaujatý vůči žádné etnické skupině.
- Utilitaristický přístup: Nasazení modelu, který optimalizuje veřejné zdraví, i když existují určité nespravedlivé výsledky vůči menšinám.

Q13.2: List at least two potential ethical problems related to data collection. [5]

- Neinformovaný souhlas: Uživatelé nemusí být plně informováni o tom, jak budou jejich data shromažďována a využívána.
- Zaujatá data: Data mohou být sbírána způsobem, který zvýhodňuje jednu skupinu a znevýhodňuje jinou, což vede k diskriminaci.

Q13.3: List at least two potential ethical problems that can originate in model evaluation. [5]

- Nezohlednění všech zúčastněných stran: Model může být hodnocen pouze z hlediska jedné skupiny uživatelů, což vede k nespravedlivým výsledkům pro jiné skupiny.
- Nesprávné metriky: Používání nevhodných nebo nedostatečných metrik může zakrýt škodlivé účinky modelu.

Q13.4: List at least one example of an ethical problem that can originate in model design or model development. [5]

- Zaujatost v návrhu: Vývojáři mohou neúmyslně vložit své vlastní předsudky do návrhu modelu, například zvolením proměnných, které nevhodně reprezentují určité populace.

Q13.5: Under what circumstances could train-test mismatch be an ethical problem? [5]

Train-test mismatch může být etickým problémem, pokud trénovací data neodpovídají skutečné populaci nebo reálnému prostředí, ve kterém bude model nasazen. To může vést k nepřesným nebo nespravedlivým výsledkům, zejména pokud model nesprávně predikuje pro marginalizované skupiny.