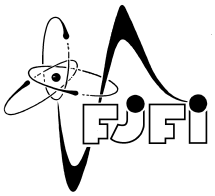
České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská



Optimalizační metody ve strojovém učení

Optimization methods in machine learning

Bakalářská práce

Autor: **Ilona Ivantsova**

Vedoucí práce: **Ing. Václav Mácha**

Akademický rok: 2022/2023

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

Katedra: matematiky

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Ilona Ivantsova
Studijní program:	Aplikace přírodních věd
Studijní obor:	Matematické inženýrství
Studijní zaměření:	Aplikované matematicko-stochastické metody
Název práce (česky):	Optimalizační metody ve strojovém učení
Název práce (anglicky):	Optimization methods in machine learning

Pokyny pro vypracování:

1) Seznamte se se základními klasifikačními modely jako jsou například logistická regrese, SVM nebo neuronové sítě. Pozornost věnujte vlastnostem jednotlivých modelů a jejich běžnému použití.
2) Seznamte se se základními optimalizačními metodami využívanými pro řešení klasifikačních modelů. Pozornost věnujte metodám založeným na gradientu jako jsou například Newtonova metoda, metoda gradientního spádu a tzv. 'stochastic gradient descent'.
3) Na vhodně zvoleném příkladu demonstrujte klady a zápory vybraných optimalizačních metod a proveďte jejich porovnání.
4) Zvolte klasifikační problém a sestavte matematický model tohoto problému. Vyberte a naprogramujte vhodnou optimalizační metodu pro řešení sestaveného modelu. Proveďte vyhodnocení kvality výsledného modelu.

Doporučená literatura:

1) T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning: data mining, inference, and prediction. Springer, New York, 2nd ed, 2009.
2) J. Nocedal, S. J. Wright, Numerical optimization. Springer, Science & Business Media, 2006.
3) C. M. Bishop, Pattern recognition and machine learning. Springer, Science & Business Media, 2006.
4) S. Boyd, L. Vandenberghe, Convex optimization. Cambridge university press, 2004.

Jméno a pracoviště vedoucího bakalářské práce:

Ing. Václav Mácha
Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze, Trojanova 13, 120 00 Praha 2

Jméno a pracoviště konzultanta:

Mgr. Lukáš Adam, Ph.D.
Ústav teorie informace a automatizace, AV ČR v.v.i., Pod Vodárenskou věží 4, 182 00 Praha 8

Datum zadání bakalářské práce: 31.10.2021

Datum odevzdání bakalářské práce: 7.7.2022

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 12. října 2021

.....
garant oboru

.....
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta jaderná a fyzikálně inženýrská -1-

.....
děkan

Poděkování:
Chtěla bych zde poděkovat především svému školiteli Václavu Máche za pečlivost, ochotu, vstřícnost a odborné i lidské zázemí při vedení mé diplomové práce. Dále děkuji svému konzultantovi Lukášu Adamovi.

Čestné prohlášení:
Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 5. ledna 2023 Ilona Ivantsova

Název práce: **Optimalizační metody ve strojovém učení**

Autor: Ilona Ivantsova

Obor: Matematické inženýrství

Zaměření: Aplikované matematicko-stochastické metody

Druh práce: Bakalářská práce

Vedoucí práce: Ing. Václav Mácha, Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze, Trojanova 13, 120 00 Praha 2

Abstrakt:

V této práce analyzujeme vztah mezi problémem klasifikace a optimalizací. Diskutujeme řadu optimalizačních algotimů, používaných pro hledání minima funkce, které jsou použitelné pro řešení klasifikačních problemů. Seznamujeme se s teorií hledání minima funkce bez omezení, zavádíme pojmy globálního a lokálního minima a diskutujeme jejich vztah s gradientem funkce. Seznamujeme se s řadou základních optimalizačních iteračních algoritmů, založených na gradientu účelové funkce. Popisujeme klasifikační model logistické regrese a SVM (z anglického *Support Vector Machines*). Ukazujeme aplikaci modelu logistické regrese na dataset Iris. Dále ukazujeme jak výběr optimalizačního algoritmu pro vyřešení klasifikačního modelu ovlivňuje počet správně a špatně klasifikovaných pozorování. Vizualizujeme trajektorije řešení jednotlivých optimalizačních algoritmů, porovnáváme jejich účinnost a diskutujeme jejich klady a zápory. Výsledné matematické modely porovnáváme a srovnáváme jejich účinnost v závislosti na použité optimalizační metodě.

Klíčová slova: gradient descent, logistická regrese, Newton-Raphson, optimalizace, SVM

Title: **Optimization methods in machine learning**

Autor: Ilona Ivantsova

Abstract:

In the following paper, we analyze the relationship between the problem of classification and optimization. We discuss a number of optimization algorithms that can be used to find the minimum value of objective function and are useful for solving classification problems. We get acquainted with the theory of unconstrained optimalization, introduce the concepts of global and local minima and discuss their relationship with the gradient of the function. We become familiar with a few basic optimization iteration algorithms, based on the gradient of the objective function. We describe the classification model of logistic regression and SVM (Support Vector Machines). We show its application to the Iris dataset and show how the selection of optimization algorithm affects the number of correctly and incorrectly classified observations. We visualize the trajectories of solutions for different optimizers, compare their efficiency, and discuss their pros and cons. Finally, we compare final mathematical models, and discuss their efficiency depending on the optimization method used.

Key words: gradient descent, logistic regresion, Newton-Raphson, optimization, SVM

Obsah

Úvod

V této práci se zabýváme vztahem klasifikace a otimalizace. Klasifikace je problém, kdy se snažíme pomocí matematického modelu zařadit pozorování x do správné třídy y. Klasifikační modely jsou obvykle definovány jako minimalizační problémy, kde se snažíme minimalizovat chyby v zařazení jednotlivých pozorování do tříd. V práci představíme několik obecných optimalizačních algoritmů pro hledání minima funkce, které lze použít pro řešení klasifikačních modělů. Dále v práci ukážeme, jak výběr optimalizačního algoritmu ovlivňuje kvalitu výsledného klasifikačního modelu a tedy i počet správně a špatně klasifikovaných pozorování.

Tak, pro klasifikátor logistické regresi chceme buď minimalizovat nebo-li maximalizovat pravděpodobnost toho, že prvek náleží do určité třídy, pro SVM chceme nalézt dvě paralelní nadroviny , které oddělují dvě třídy a jejich vzdálenost od sebe je co největší.

V první části prace se seznámíme s teorií hledání minima funkce bez omezení a zavedeme základní pojmy jako jsou lokální a globální minimum funkce a jejich vztah ke gradientu funkce. Ukazujeme, že optimalizační metody jsou většinou založené na gradientu funkce a proto jsou schopné hledat pouze minima lokální (resp. stacionární body) a nezaručují nalezení minima globálního. Dále diskutujeme důležitou třídu konvexních funkcí, jelikož pro konvexní funkce platí, že bod lokálního optimuma je zároveň bodem optima globálního a proto je výhodnější optimalizovat funkce konvexní.

Ve druhé kapitole se seznamujeme se základními gradientními iteračními optimalizačními algoritmy, které lze mimo jiné využít pro řešení klasifikačních modelů. Nejprve popisujeme metodu gradientního spádu (gradient descent) a její varianty, např. momentum gradient descent, Nesterov gradient descent nebo metoda ADAM. Také popisujeme stochastickou verzi metody gradient descent. Jako další představujeme gradientní metodou druhého řádu Newton-Raphson.Na příkladě Baelovy funkce ukazujeme jak jsou všechny popsané metody účinné a diskutujeme jejich klady a zápory. Pro porovnání používáme vizualizaci trajektorie řešení v iteracích algoritmu a také hodnotu minimalizované funkce v jednotlivých iteracích algoritmu.

Ve třetí kapitole popisujeme model logistické regrese a způsob jak tento klasifikátor hledá separační nadrovinu, která rozděluje prvky do jednotlivých třídd.

Ve čtvrté kapitole popisujeme liniarní klasifikátor SVM (z anglického *Support Vector Machines*). Ukazujeme jak základní verzi pro dokonale lineárně separovatelná data tak i variantu pro data, která lineárně separovatelná nejsou. Nakonec popisujeme tzv. jádrový trik, který umožňuje díky projekci do vyšší dimenze SVM znelinearizovat a použít i na data, která jsou lineárně neseparovatelná.

V páté kapitole ukazujeme použití modelu logistické regrese na skutečná data. Jako data jsme zvolili dataset *Iris*, který ma za cíl klasifikovat na základě šířky a délky kališního a okvětního lístku jednotlivá pozorování jako jeden ze tří druhů kosatců. Pro vyřešení modelu logistické

regrese jsme použili jednotlivé optimalizační algoritmy definované ve druhé kapitole. Výsledné modely jsme poté porovnali a srovnali jak se liší v závislosti na pužitém optimalizátoru.

Kapitola 1

Minimalizace funkce bez omezení

Cílem optimalizace je buď minimalizovat nebo maximalizovat funkci f na nějaké množině $\Theta \subset \mathbb{R}^d$, která je podmnožinou d-rozměrného Eukleidovského prostoru

$$\begin{aligned} \max \quad & f(\boldsymbol{\theta}), \\ \text{s. t.} \quad & \boldsymbol{\theta} \in \Theta, \end{aligned} \tag{1.1}$$

kde $\boldsymbol{\theta} \in \Theta$ je vektor proměnnych (parametrů) dimenze $d \geq 1$ a $f : \mathbb{R}^d \to \mathbb{R}$ je skalární funkce, závislá na $\boldsymbol{\theta}$, kterou chceme v tomto případě maximalizovat. Nemusíme ale uvažovat oboje maximalizační a minimalizační problémy. V podstatě se jedná o ten stejný problém a platí maximalizační problém

$$\max \quad f(\boldsymbol{\theta}),$$

můžeme snadno převést na ekvivalentní problém minimalizační

$$-\min \quad -f(\boldsymbol{\theta}).$$

Jinak řečeno: vektor $\boldsymbol{\theta}^\star$ maximalizuje funkci f pravě tehdy, když minimalizuje funkci $-f$. Tudíž pokud chceme řešit maximalizační problém budeme místo funkce f uvažovat funkci $-f$ a takovou funkci budeme minimalizovat.Tento trik má za důsledek, že všechny numerické a teoretické výsledky stačí odvodzovat pouze pro problém minimalizace.

Optimalizace bez omezeni, je speciálním případem optimalizace, kde optimalizujeme funkci f na celém prostoru $\Theta = R^d$. Tímto se optimalizační problém (1.1) zredukuje na následující tvar

$$\max_{\boldsymbol{\theta} \in \mathbb{R}^d} \quad f(\boldsymbol{\theta}), \tag{1.2}$$

kde $\boldsymbol{\theta} \in \mathbb{R}^d$ je vektorem reálných čísel dimenze $d \geq 1$ a $f : \mathbb{R}^d \to \mathbb{R}$ je skalární funkce definovaná na celém $\mathbb{R}^d$.

1.1 Gobální a lokální minima

Pro úplnosty definice optimalizačního problému bez omezení je třeba formálně definovat co je to minimum funkce.

Definice 1 (Globální minimum [5])**.** *Nechť je funkce* $f(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ *definovaná na množině* $\Theta \subset \mathbb{R}^d$ *a nechť* $\boldsymbol{\theta}^\star \in \Theta$*. Řekneme, že funkce* f *má v bodě* $\boldsymbol{\theta}^\star$ ***globální minimum*** *na množině* Θ*, jestliže pro každé* $\boldsymbol{\theta} \in \Theta \setminus \{\boldsymbol{\theta}^\star\}$ *platí*

$$f(\boldsymbol{\theta}) \geq f(\boldsymbol{\theta}^\star).$$

Řekneme, že funkce f *má v bodě* $\boldsymbol{\theta}^\star$ ***ostré globální minimum*** *na množině* Θ*, jestliže pro každé* $\boldsymbol{\theta} \in \Theta \setminus \{\boldsymbol{\theta}^\star\}$ *platí*

$$f(\boldsymbol{\theta}) > f(\boldsymbol{\theta}^\star).$$

Předchozí definice tedy říká, že $\boldsymbol{\theta}^\star$ minimalizuje funkci f, pokud na celém prostoru Θ není žádný bod, ve kterém by měla funkce f menší hodnotu než v bodě $\boldsymbol{\theta}^\star$. Hledání globálních minim je velice složité a většina numerických algoritmů je navržena pouze na hledání minim lokálních. Důvodem je, že zpravidla můžeme funkci f vyhodnotit pouze v konečném počtu bodů a tak nemáme dobrou představu o tom, jak funkce vypadá celkově a nemůžeme si být jisti, že v nějaké oblasti, kterou algoritmus neuvažuje, nemá funkce minimum globální.

Definice 2 (Lokální minimum [5])**.** *Nechť je funkce* $f(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ *definovaná na množině* $\Theta \subset \mathbb{R}^d$ *a nechť* $\boldsymbol{\theta}^\star \in \mathrm{der}(\Theta)$*. Řekneme, že funkce* f *má v bodě* $\boldsymbol{\theta}^\star$ ***lokální minimum*** *na množině* Θ*, jestliže existuje redukované okolí* $U^*(\boldsymbol{\theta}^\star)$ *takové, že pro každé* $\boldsymbol{\theta} \in U^*(\boldsymbol{\theta}^\star) \cap \Theta$ *platí*

$$f(\boldsymbol{\theta}) \geq f(\boldsymbol{\theta}^\star).$$

Řekneme, že funkce f *má v bodě* $\boldsymbol{\theta}^\star$ ***ostré lokální minimum*** *na množině* Θ*, jestliže existuje redukované okolí* $U^*(\boldsymbol{\theta}^\star)$ *takové, že pro každé* $\boldsymbol{\theta} \in U^*(\boldsymbol{\theta}^\star) \cap \Theta$ *platí*

$$f(\boldsymbol{\theta}) > f(\boldsymbol{\theta}^\star).$$

Samozřejmé platí, že každé globální minimum je zároveň i minimem lokálním. Proto je vždy lepší hledat minima globální. Vztah mezi globálním a lokálním minimem je znázorněn na obrázku 1.1.

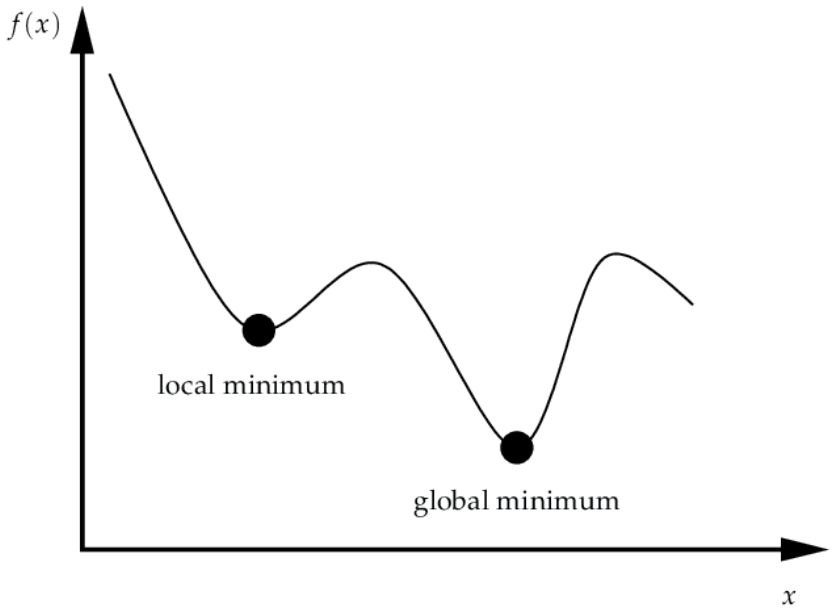
1.2 Vztah minimalizace a gradientu

Z definice 2 se může zdát, že jediným způsobem jak zjistit zda je bod $\boldsymbol{\theta}$ bodem lokalního minima, je vyšetřit všechny body blizké tomuto bodu a přesvědčit se, že v žádném z těchto bodů nema funkce f menší funkčí hodnotu. Nicméně tento způsob je v praxi neproveditelný. Pokudje je funkce f spojitě deferencovatelná, tak můžeme hledat lokální minima vyšetřováním jejího gradientu a hessiánu.

Definice 3 (Gradient funkce [5])**.** *Nechť je dána funkce* $f(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ *definovaná na množině* $\Theta \subset \mathbb{R}^d$*. Gradientem* ∇f *funkce* f *na množině* Θ *rozumíme funkční vektor*

$$\nabla f(\boldsymbol{\theta}) := \left(\frac{\partial f}{\partial x_1}(\boldsymbol{\theta}), \ldots, \frac{\partial f}{\partial x_r}(\boldsymbol{\theta}) \right),$$

přičemž všechny výše uvedené parciální derivace existují na množině $\Theta \subset \mathbb{R}^d$.



Obrázek 1.1: Porovnaní globálního a lokálního minima.

Definice 4 (Hessova matice [5])**.** *Nechť je dána funkce* $f(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ *a nechť existují všechny její parciální derivace druhého řádu*

$$\frac{\partial f^2}{\partial x_k \partial x_j}(\boldsymbol{\theta})$$

v bodě $\boldsymbol{\theta} \in \mathrm{Dom}(f)$*. Pak matici*

$$\nabla^2 f(\boldsymbol{\theta}) := \left(\frac{\partial^2 f}{\partial x_j \partial x_k}(\boldsymbol{\theta}) \right)_{j,k=1}^{d}$$

budeme nazývat ***Hessovou maticí*** *funkce* f *v bodě* $\boldsymbol{\theta}$*. Tj. prvkem Hessovy matice vyskytujícím se na* j*-tém řádku a* k*-tém sloupci je parciální derivace*

$$\frac{\partial f^2}{\partial x_k \partial x_j}(\boldsymbol{\theta}).$$

Definice 5 (Stacionární bod [5])**.** *Bod* $\boldsymbol{\theta} \in \mathbb{R}^d$ *nazveme stacionarním bodem funkce* $f(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$*, jestliže platí* $\nabla f(\boldsymbol{\theta}) = \mathbf{0}$*.*

Zaměřme se tedy spojitě deferencovatelné funkce. Potom můžeme pro hledání jejích lokálních minim použít následující větu.

Věta 1 (Nutná podmínka pro lokalní extrem [7])**.** *Nechť je dána funkce* $f(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$*. Nechť* $\boldsymbol{\theta}^\star \in \mathbb{R}^d$ *je lokálním minimem funkce* f *a funkce* f *je spojitě deferencovatelná na otevřeném okolí bodu* $\boldsymbol{\theta}^\star$*. Potom platí*

$$\nabla f(\boldsymbol{\theta}^\star) = \mathbf{0}.$$

Nutná podminka pro lokalní minimum říká, že každy bod lokalniho minima je zároveň stacionarním bodem. Podle poslední úvahy a podle definice gradientu je zřeme, že má-li funkce v bodě $\boldsymbol{\theta}$ lokální minimum, pak její gradient je v tomto bodě nulový. Numerické algoritmy, které se používají pro optimalizaci, nehledají globální ani lokální minima ale pouze stacionární body. Důvodem je skutečnost, že nalezení bodu ve kterém je nulový gradient je daleko snažší úloha.

Důležitou třídou funkcí pro optimalizaci jsou funkce konvexní.

Definice 6 (Konvexní funkce [7])**.** *Funkce* $f(\boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}$ *je konvexní, pokud pro všechny dvojice bodů* $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathrm{Dom}(f)$ *a každé* $\alpha \in [0, 1]$ *platí*

$$f(\alpha\boldsymbol{\theta}_1 + (1 - \alpha)\boldsymbol{\theta}_2) \leq \alpha f(\boldsymbol{\theta}_1) + (1 - \alpha) f(\boldsymbol{\theta}_2).$$

Důvodem je, že pro tyto funkce jsme schopni efektivně nalézt i globální minimum a to proto, že platí následující věta.

Věta 2 ([7])**.** *Pokud* f *je konvexni funkce, tak každé lokalní minimum* $\boldsymbol{\theta}^\star$ *je zároveň minimem globálním. Pokud je navíc* f *je diferencovatelná, pak taky stacionarni bod je minemem globálním.*

Kapitola 2

Numerické metody pro hledání minima

V předchozí kapitole jsme představili problém optimalizace bez omezení a ukázali vztah mezi hledáním minima funkce a gradientem. V této kapitole představíme dvě numerické algoritmy založené na gradientu funkce, které se běžně používají pro řešení uvedeného optimalizačního problému: metoda gradientního spádu [9] a metodu Newton-Raphson [1]. Oba tyto algoritmy jsou tzv. iterační algoritmy. Hledání minima začíná stanovaním počátečního řešení $\boldsymbol{\theta}_0$ a poté se tyto algoritmy snaží postupně po jednotlivých krocích (iteracích) toto počáteční řešení zlepšovat. Algoritmus je obvykle zakončen pokud nalezne řešení, které splňuje nějakou ukončující podmínku, nebo pokud je dosažen maximální možný počet iterací. Způsob, jakým je v jednotlivých iteracích vylepšováno řešení je specifický pro každý algoritmus. Algoritmy obvykle pro vylepšování řešení používají hodnoty funkce f, její gradient ∇f a případně Hessovu matici $\nabla^2 f$. Některé z algoritmů navíc akumulují informace nasbírané během předešlých iterací, čímž se snaží zrychlit konvergenci. Nezávisle na tom jakým způsobem daný algoritmus vylepšuje řešení, každý kvalitní algoritmus by měl splňovat následující vlastnosti:

1. **Robustnost:** algoritmus musí být schopen řešit rozmanité problémy a měl by být co nejméně náchylný na volbu počátečního řešení.

2. **Efektivita:** algoritmus by měl být rozumně náročný a to jak časově tak i výpočetní náročností.

3. **Přesnost:** algoritmus by měl být schopen dosahovat dostatečně přesného řešení.

Tyto vlastnosti mohou jít proti sobě, např. může existovat algoritmus, který je sice velice rychlý, ale vyžaduje pro výpočet hodně paměti a proto takovou metodu nelze použít na problémy s vysokou dimenzí. Dalé velice často platí, že robustní metody jsou pomalejší. Je tedy třeba vždy zvolit vhodnou metodu pro daný problém a to v závislosti na tom jaké jsou požadavky na rychlost výpočtu a přesnost řešení.

2.1 Metoda gradient descent

Jednou z nejběžněji používaných metod pro hledání minima funkce je metoda gradientního spádu [9]. Často se také můžeme setkat s pojmenováním metoda gradient descent. Jak již bylo

řečeno výše, jedná se o iterační alogitmus, který lze popsat následovně. Předpokládejme, že máme funkci $f(\boldsymbol{\theta})$, $f : \mathbb{R}^d \to \mathbb{R}$, která je diferencovatelná vůči svým parametrům $\boldsymbol{\theta}$. Prvním krokem pro nalezení minima této funkce je volba počátečního odhadu řešení $\boldsymbol{\theta}_0$.. Po zvolení vhodného počátečního odhadu řešení opakujeme následující kroky:

1. Nalezne směr sestupu $\boldsymbol{d}_k \in \mathbb{R}^d$, tj. směr, který povede na zmenšení hodnoty funkce f a tedy na zlepšení řešení.
2. Zvolíme délku kroku $\alpha_k \in \mathbb{R}$.
3. Spočteme nové řešení $\boldsymbol{\theta}_{k+1}$ podle následujícího vzorce

$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + \alpha_k \cdot \boldsymbol{d}_k \tag{2.1}$$

4. Zkontrolujeme, zda nové řešení $\boldsymbol{\theta}_{k+1}$ splňuje podmínku ukončení. Pokud ano, tak jsme dosáhli řešeni s požadovanou přesností. Pokud ne, tak se vracíme na první bod a pokračujeme v hledání.

Výše uvedený algoritmus je obecný algoritmus pro hledání minima funkce a nazývá se *Descent Direction Iteration*. Cílem algoritmu je to, aby vždy platilo

$$f(\boldsymbol{\theta}_k) \geq f(\boldsymbol{\theta}_{k+1}).$$

Pro dosažení této nerovnosti je třeba zvolit vhodný směr sestupu $\boldsymbol{d}_k$ a také vhodnou délku kroku α_k. Špatnou volbou délky kroku se může velice snadno stát, že i pokud použijeme správný směr sestupu, tak současné řešení nezlepšíme, ale zhoršíme.

2.1.1 Vanilla gradient descent

Descent Direction Iteration je obecný algoritmus a (Vanilla) gradient descent [9] je speciální případ, kdy jako směr volíme minus gradient funkce f tzn.

$$\boldsymbol{d}_k = -\nabla f(\boldsymbol{\theta}_k)$$

připadne se občas užívá normalizovaná verze

$$\boldsymbol{d}_k = -\frac{\nabla f(\boldsymbol{\theta}_k)}{\|\nabla f(\boldsymbol{\theta}_k)\|}$$

Důvod k teto volbě je jednoduchý: gradient udává směr největšího růstu funkce a tedy pokud chceme funkci minimalizovat, tak dává smysl použít směr opačný. Volbou tohoto směru můžeme přepsat rovnici (2.1) na následující tvar

$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \alpha_k \cdot \nabla f(\boldsymbol{\theta}_k)$$

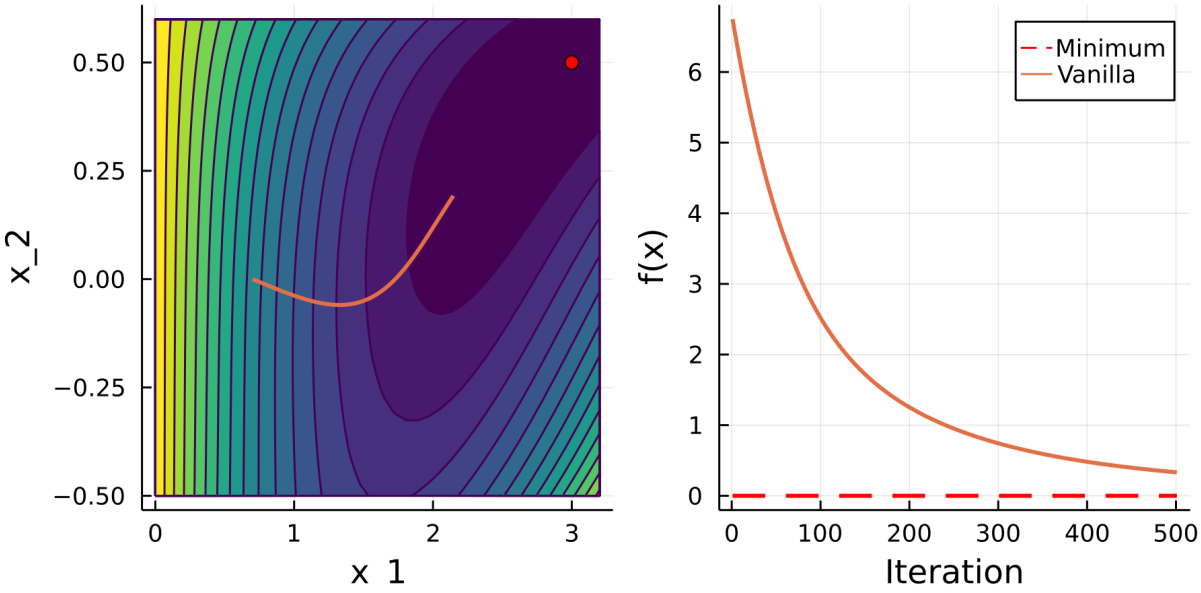
kde α_k je iterační krok, který určuje, jak velké kroky ve zvoleném směru budeme dělat. Hodnotu α_k musíme vybírat rozumně, aby byla zajištěna konvergence a zároveň bylo dosaženo minimum funkce. V případě příliš velké α_k by mohla nastat situace, že algoritmus bude kolísat kolem

minima funkce a nedosáhne ho nikdy. Naopak, v případě malé hodnoty α_k bude algoritmus velice pomalý a nemusí dosáhnout minima ve stanoveném maximálním počtu iteraci.

Pro ilustraci fungování výše popsaného algoritmu uvažujme následující tzv. Baelovu funkci

$$f(\theta_1, \theta_2) = (1.5 - \theta_1 + \theta_1 \cdot \theta_2)^2 + (2.25 - \theta_1 + \theta_1 \cdot \theta_2^2)^2 + (2.625 - \theta_1 + \theta_1 \cdot \theta_2^3)^2 \tag{2.2}$$

Naším cílem bude nalézt minimum této funkce pro $-4.5 \leq \theta_1, \theta_2 \leq 4.5$. Na této množině má uvedená funkce minimum v bodě $\boldsymbol{\theta}^\star = [3, 0.5]$ a funkční hodnota v tomto bodě je $f(3, 0.5) = 0$. Jako počáteční bod pro uvedený algoritmus volíme $\boldsymbol{\theta}^\star = [0, 0]$, délku kroku volíme konstantní $\alpha = \alpha_k = 0.001$ a maximální počet iterací stanovujeme na 500. V levé části obrázku 2.1 vidíme trajektorii jak se postupně v iteracích zlepšovalo řešení nalezené pomocí vanilla gradient descentu. Je zřejmé, že ani po 500 iteracích algoritmus nedosáhl optimálního řešení. Důvodem byla volba příliš krátkého kroku, která zapříčinila pomalou konvergenci metody. Nícméně z pravé části obrázku 2.1 je zřejmé, že funkční hodnota se v průběhu iterací neustále zmenšovala a tedy algoritmus konvergoval ke správnému řešení.



Obrázek 2.1: Vývoj trajektorie (levá část) a funkční hodnoty (pravá část) při hledání minima Baelovy funkce (2.2) pomocí metody vanilla gradient descent. Červený bod a červená přerušovaná čára predstavují optimální řešení a optimální funkční hodnotu.

2.1.2 Momentum

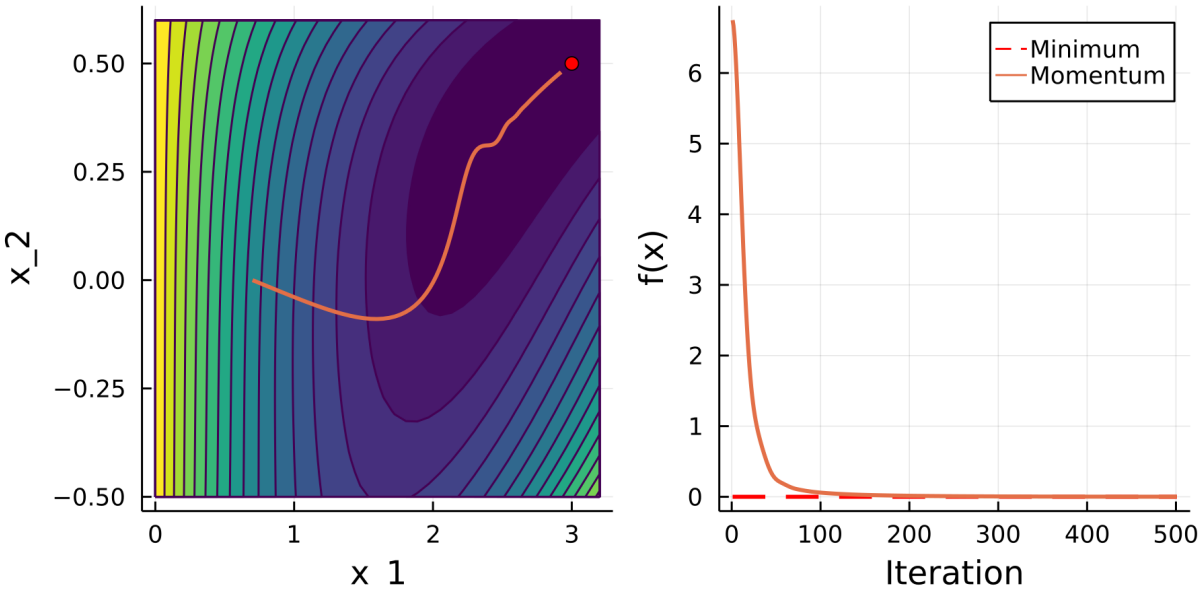
Jak je vidět z obrázku 2.1, metoda gradient descent je málo efektivní v případech, kdy se pohybuje delší dobu stejným směrem. Důvodem je, že metoda neadaptuje délku kroku v případě, že se pohybuje delší dobu stejným směrem a tedy je v takové situaci "pomalá". Pro zlepšení tohoto chování byla vyvinuta metoda momentum [8]. Hlavní myšlenku metody momentum si lze představit následovně: uvažujme že máme míč a pustíme ho dolů z kopce do údolí. Míč se

bude pohybovat pořád stejným směrem, ale gravitace bude způsobovat jeho zrychlování. Metoda momentum se snaží dosáhnout tohoto zrychlování. Abychom tohoto mohli dosáhnout, tak musíme byt schopni použít při aktualizaci řešení nejen současný gradient, ale také nedávnou historii změn parametrů. Pro metodu momentum můžeme přepsat rovnici (2.1) na následující tvar

$$v_k \leftarrow \gamma \cdot v_{k-1} + \alpha \cdot \nabla f(\boldsymbol{\theta}_k),$$
$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - v_k,$$

kde parametr γ je zpravidla nastaven na konstantní hodnotu 0.9. Přičemž pro $\gamma = 0$ znovu dostáváme model, odpovídající metodě Vanilla gradient descent.

Pro ilustraci fungování opět použijeme Baelovu funkci (2.2). V levé části obrázku 2.2 vidíme trajektorii jak se postupně v iteracích zlepšovalo řešení nalezené pomocí momentum gradient descentu. Je zřejmé, že oproti metodě vanilla gradient descent dosáhla tato metoda lepšího řešení. Ani tato metoda ovšem nedosáhla řešení optimálního. Nícméně z pravé části obrázku 2.2 je zřejmé, že funkční hodnota se v průběhu iterací neustále zmenšovala a tedy i tento algoritmus konvergoval ke správnému řešení. Také si můžeme všimnout, že pokles funkční hodnoty je daleko rychlejší než v případě metody Vanilla gradient descent.



Obrázek 2.2: Vývoj trajektorie (levá část) a funkční hodnoty (pravá část) při hledání minima Baelovy funkce (2.2) pomocí metody momentum gradient descent. Červený bod a červená přerušovaná čára predstavují optimální řešení a optimální funkční hodnotu.

2.1.3 Nesterov gradient descent

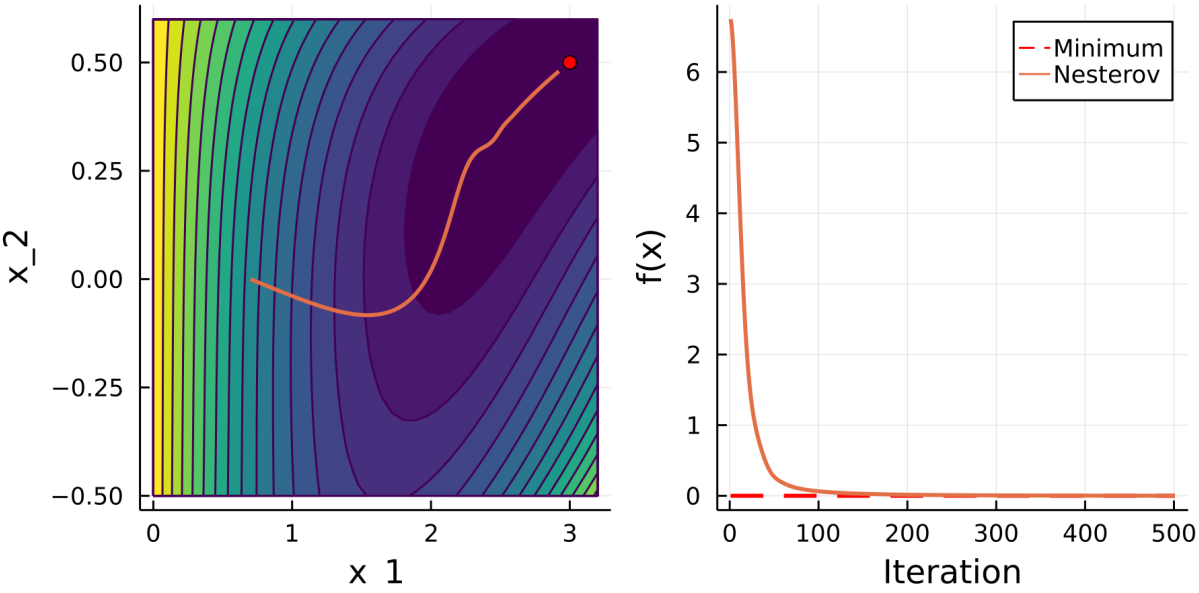
Potíži metody momentum může vést na situaci, že metoda přeskočí hledané minimum v dusledku naakumulované rychlosti z předchozích iteračních kroků. Pokud se vrátíme k příkladu

s míčem, tak by tato situace odpovídala situaci, kdy míč z kopce zrychlí tolik, že přejede nejnižší bod údolí a vyjede na druhé straně údolí kus do kopce. Nesterova metoda [6] upravuje metodu momentum, tak, aby k takové situaci nedocházela. Úprava spočívá v tom, že metoda nepočítá gradient v současném řešení, ale počítá gradient v očekávaném budoucím řešení. Pro metodu nesterov můžeme přepsat rovnici (2.1) na následující tvar

$$\begin{aligned} v_k &\leftarrow \gamma \cdot v_{k-1} + \alpha \cdot \nabla f(\boldsymbol{\theta}_k - \gamma \cdot v_{k-1}), \\ \boldsymbol{\theta}_{k+1} &\leftarrow \boldsymbol{\theta}_k - v_k, \end{aligned}$$

kde parametr γ je zpravidla nastaven na konstantní hodnotu 0.9. Přičemž pro $\gamma = 0$ znovu dostáváme opět model, odpovídající metodě Vanilla gradient descent.

Pro ilustraci fungování opět použijeme Baelovu funkci (2.2). V levé části obrázku 2.3 vidíme trajektorii jak se postupně v iteracích zlepšovalo řešení nalezené pomocí momentum gradient descentu. Trajektorie je velice podobná té pro metodu momentum. Obdobně z pravé části obrázku 2.2 je zřejmé, že funkční hodnota se v průběhu iterací neustále zmenšovala a to velice podobně jako v případě metody momentum.



Obrázek 2.3: Vývoj trajektorie (levá část) a funkční hodnoty (pravá část) při hledání minima Baelovy funkce (2.2) pomocí metody Nesterov gradient descent. Červený bod a červená přerušovaná čára predstavují optimální řešení a optimální funkční hodnotu.

2.1.4 ADAM

Metoda momentum a metoda nesterov aktualizují všechny parametry $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_d)$ současně a používájí pro všechny parametry stejnou hodnotu délky kroku v každé iteraci. Metoda ADAM [4] (anglicky *Adaptive Moment Estimation*), na rozdíl od výše uvedených, přizpůsobuje iterační krok α_k každému parametru $\theta_1, \theta_2, \ldots, \theta_d$ zvlášť a navíc upravuje délku kroku

α_k. Metoda nejprve spočte následující pomocné proměnné

$$m_k \leftarrow \beta_1 \cdot m_{k-1} + (1-\beta_1)\nabla f(\boldsymbol{\theta}_k)$$
$$v_k \leftarrow \beta_2 \cdot v_{k-1} + (1-\beta_2)\nabla f(\boldsymbol{\theta}_k)^2$$

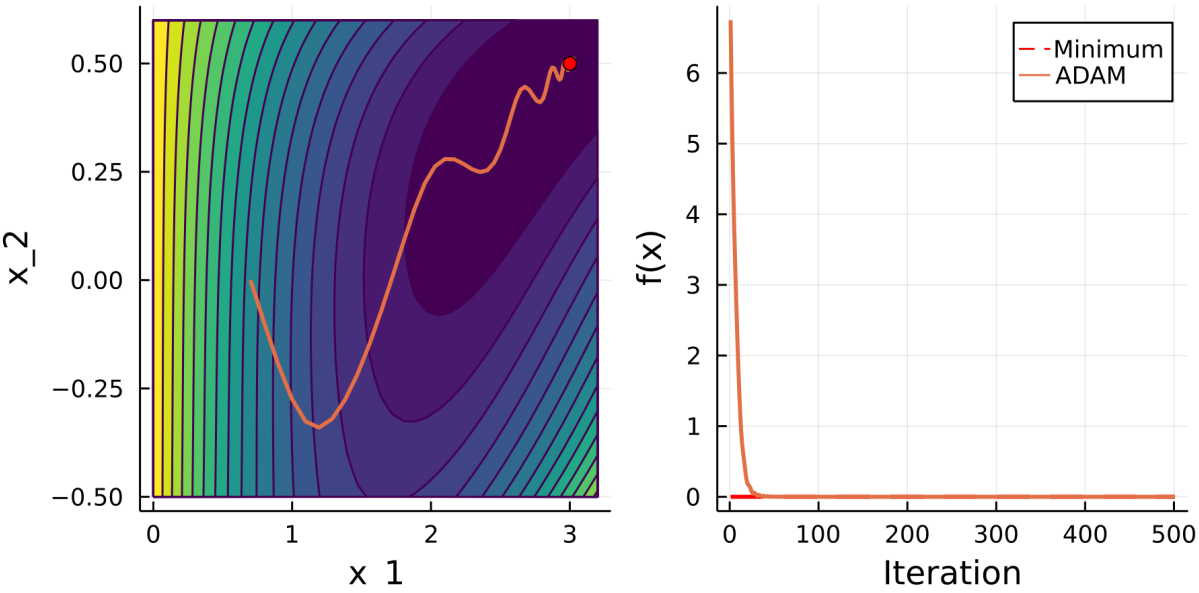
Proměnná m_k představuje odhad prvního momentu gradientu (průměru) a proměnná v_k představuje odhad druhého momentu (rozptylu) gradientu. Pro metodu ADAM můžeme přepsat rovnici (2.1) na následující tvar

$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \frac{\alpha}{\sqrt{\hat{v}_k} + \varepsilon}\hat{m}_k,$$

kde

$$\hat{m}_k = \frac{v_k}{1-\beta_1^k}, \qquad \hat{v}_k = \frac{m_k}{1-\beta_2^k}.$$

Pro ilustraci fungování opět použijeme Baelovu funkci (2.2). V levé části obrázku 2.4 vidíme, že trajektorii řešení pro metodu ADAM se výrazně liší od trajektorií předchozích. Nicméně tato metoda jako jediná dokonvergovala do optimálního řešení v 500 iteracích. Z pravé části obrázku 2.4 je také vidět, že funkční hodnota se v průběhu iterací neustále zmenšovala a ze všech uvedených metod klesala nejrychleji.



Obrázek 2.4: Vývoj trajektorie (levá část) a funkční hodnoty (pravá část) při hledání minima Baelovy funkce (2.2) pomocí metody ADAM. Červený bod a červená přerušovaná čára představují optimální řešení a optimální funkční hodnotu.

2.1.5 Stochastická metoda gradient descent

U všech výše uvedených modificaí metody gradient descent jsme uvažovali, že známe přesný předpis funkce, kterou se snažíme minimalizovat. Nicméně v praxi se velice často dostáváme do situace, že funkce f je závislá nejen na parametrech $\boldsymbol{\theta}$, ale také na nějaké sadě dat

$$\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n},$$

kde $n \in \mathbb{N}$ je počet vzorků v množině $\mathcal{T}$. Tato množina dat udává tvar funkce a při výpočtu gradientu funkce $f(\boldsymbol{\theta}) = f(\boldsymbol{\theta}; \mathcal{T})$ ji považujeme za konstantu. V případě, že je tato množina malá, tak můžeme snadno spočítat gradient funkce f a ve výpočtu uvažovat celou množinu $\mathcal{T}$. Typicky je ovšem množství dat popisující funkci f tak veliké, že nejsme schopni počítat gradient na celých datech najednou. V takovém případě musíme přistoupit k nějaké aproximaci této funkce.

Nejrozšířenejším přístupem pro vyřešení uvedeného problému je použití metody stochatic gradient descent [2], která je někdy také nazývana mini-batch gradient descent. Tato metoda nahrazuje při výpočtu gradientu a aktualizaci řešení množinu $\mathcal{T}$ její podmnožinou, která je v každé iteraci náhodně vybrána (proto stochastic). Ve výsledku používáme celá data, ale ne najednou. To tedy znamená, že v každé iteraci nejprve náhodně vybereme podmnožinu $\mathcal{T}_{mini}$ množiny $\mathcal{T}$ (která se nazývá mini-batch) a poté použijeme funkci

$$\hat{f}(\boldsymbol{\theta}) = f(\boldsymbol{\theta}; \mathcal{T}_{mini}) \approx f(\boldsymbol{\theta}; \mathcal{T})$$

pro aproximaci skutečné funkce f při hledání nového řešení. Ostatní kroky algoritmu stochastic gradient descent už jsou shodné s algoritmem gradient descent popsaným výše.

Nevýhodou uvedeného přístupu je, že nepoužíváme skutečnou funkci f, ale pouze její aproximaci, což může vést k špatným odhadům gradientů. Nicméně obrovskou výhodou je, že výběrem vhodné velikosti mini-batchů můžeme vyřešit i problémy, které by jinak byly z důvodů velikosti dat neřešitelné.

2.2 Newton-Raphson algorithm

Dalšímetodou, která je velice často používána pro hledání minima funkce, je metoda Newton-Raphson [1]. Oprotin metodě gradient descent ovšem tato metoda používá i derivace druhého řádu. Předpokládejme, že máme dvakrát diferencovatelnou funkci $f : \mathbb{R}^d \to \mathbb{R}$ a chceme nalézt její minimum, tzn. vyřešit následující optimalizační problém

$$\max_{\boldsymbol{\theta} \in \mathbb{R}^d} \quad f(\boldsymbol{\theta}),$$

Stejně jako metoda gradient descent, tak i metoda Newton-Raphson je iterační metoda, tzn. metoda se snaží postupně zlepšovat počáteční odhad řešení $\boldsymbol{\theta}_0$ pomocí nějakého aktualizačního pravidla. Pro nalezení pravidla pro aktualizaci současného řešení je použita aproximace funkce f pomocí Taylorova rozvoje druhého řádu v bodě $\boldsymbol{\theta}_k$

$$f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}_k) + (\boldsymbol{\theta} - \boldsymbol{\theta}_k)\nabla f(\boldsymbol{\theta}_k) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \nabla^2 f(\boldsymbol{\theta}_k)(\boldsymbol{\theta} - \boldsymbol{\theta}_k),$$

kde $\nabla f(\boldsymbol{\theta}_k)$ je gradient funkce f v bodě $\boldsymbol{\theta}_k$ a $\nabla^2 f(\boldsymbol{\theta}_k)$ je Hessova matice funkce f v bodě $\boldsymbol{\theta}_k$. Víme, že chceme funkci f minimalizovat a že v bodě minima musí splňovat $\nabla f(\boldsymbol{\theta}) = \mathbf{0}$. Dosazením dostáváme

$$\mathbf{0} = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \approx \nabla f(\boldsymbol{\theta}_k) + (\boldsymbol{\theta} - \boldsymbol{\theta}_k)\nabla^2 f(\boldsymbol{\theta}_k),$$

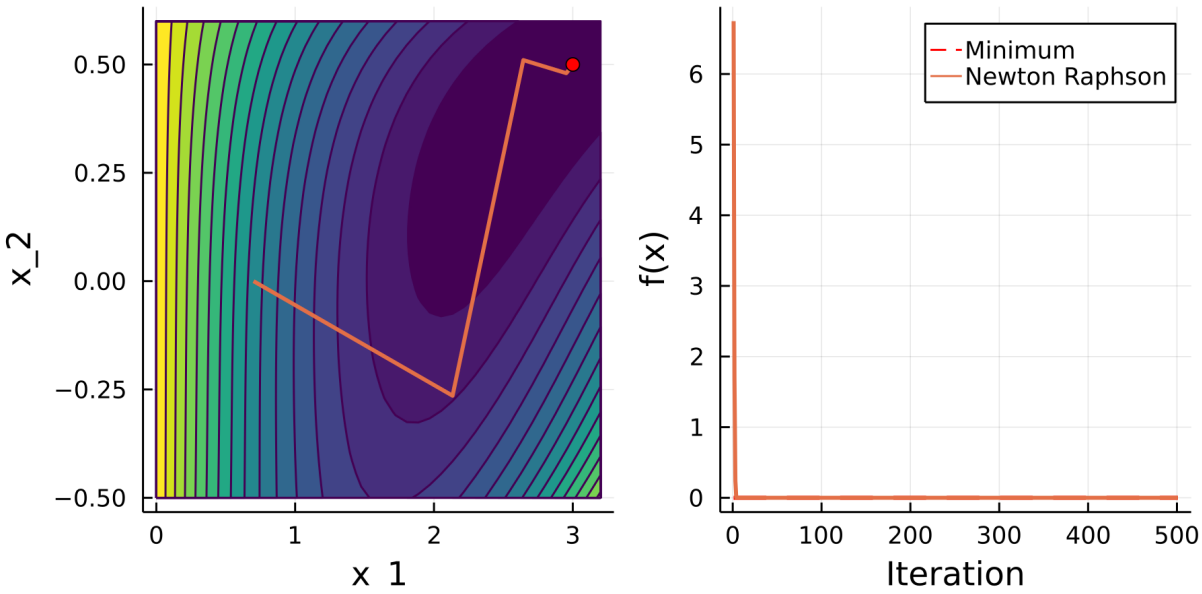
Předpokládejme, že Hessova matice je pozitivně definitní. Pak uvedená rovnost má jednoznačné řešení ve tvaru

$$\boldsymbol{\theta} = \boldsymbol{\theta}_k - (\nabla^2 f(\boldsymbol{\theta}_k))^{-1}\nabla f(\boldsymbol{\theta}_k).$$

Tímto získáváme odhad nového řešení $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}$. V praxi se navíc používá upravená verze s adaptivním hledáním délky kroku. Výsledný krok metody tedy vypadá následovně

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha_k \cdot (\nabla^2 f(\boldsymbol{\theta}_k))^{-1}\nabla f(\boldsymbol{\theta}_k).$$

Pro ilustraci fungování opět použijeme Baelovu funkci (2.2). V levé části obrázku 2.5 vidíme, že trajektorii řešení pro metodu Newton-Raphson se výrazně liší od trajektorií předchozích. Tato metoda pokud konverguje, tak k řešení koverguje velice rychle. V uvedeném příkladu metoda dokonvergovala v několika málo iteracích. Problém metody je, že je výpočetně náročnější a také více náchylná na volbu počátečního bodu.

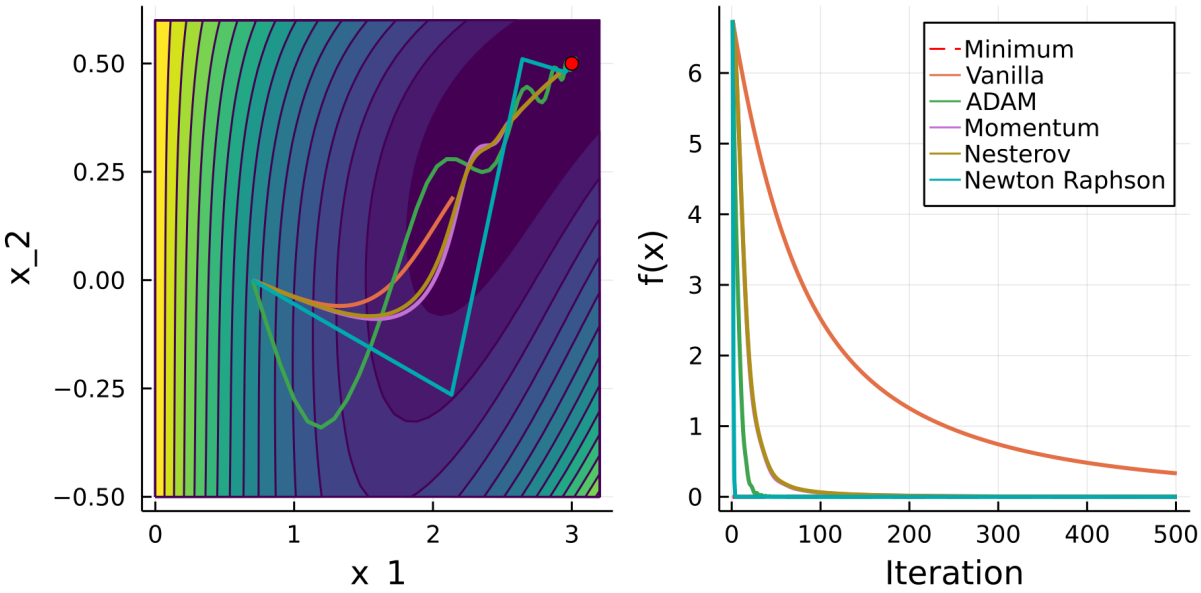


Obrázek 2.5: Vývoj trajektorie (levá část) a funkční hodnoty (pravá část) při hledání minima Baelovy funkce (2.2) pomocí metody Newton-Raphson. Červený bod a červená přerušovaná čára predstavují optimální řešení a optimální funkční hodnotu.

2.3 Porovnání

Pro lepší porovnání uvádíme ještě obrázek 2.6, který kominuje obrázky 2.1-2.4 do jednoho. Je zřejmé, že pouze metody ADAM a Newton-Raphson byly schopné dokonvergovat do optimálního řešení. Dále je vidět, že nejpomalejší konvergenci má mnetoda Vanila gradient descent.

Nicméně tyto výsledky lze výrazně ovlivnit volbou jiných parametrů pro metody. Vždy je tedy velice důležité vybrat parametry vhodné pro uvažovaný optimalizační problém, protože špatná volba parametrů může vést k výraznému zhoršení konveregcen nebo dokonce k úplnému selhání jednotlivých metod.



Obrázek 2.6: Vývoj trajektorie (levá část) a funkční hodnoty (pravá část) při hledání minima Baelovy funkce (2.2) pomocí metod Vanilla gradient descent, Momentum, Nesterov, ADAM a Newton-Raphson. Červený bod a červená přerušovaná čára predstavují optimální řešení a optimální funkční hodnotu.

Kapitola 3

Logisticka regrese

Logistická regrese [3] je klasifikační algoritmus, který se snaží aproximovat podmíněnou pravděpodobnost $\mathbf{P}(Y \mid X)$ pomocí sigmoid funkce aplikované na lineární kombinaci vztupních pozorování, tzn.

$$\mathbf{P}(Y = 1 \mid X = \boldsymbol{x}) = \sigma(z),$$

kde $z = w_0 + \sum_{i=1}^{d} w_i \cdot x_i$ a sigmoid funkce je definována následovně

$$\sigma(z) = \frac{\exp\{z\}}{1 + \exp\{z\}}$$

Pro jednoduchost se velice často používá ekvivalentní formulace v následujícím tvaru

$$\mathbf{P}(Y = 1 \mid X = \boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}),$$

kde je vektor pozorování $\hat{\boldsymbol{x}}$ rozšířen o jedničku na první pozici, tzn. $\boldsymbol{x} = (1, \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_d) \in \mathbb{R}^{d+1}$. Dále budeme pro jednoduchost uvažovat tuto formulaci. V případě binární klasifikace, tzn. v případě kdy máme pouze dvě třídy, můžeme snadno odvodit tvar pravděpodobnosti i pro druhou třídu

$$\mathbf{P}(Y = 0 \mid X = \boldsymbol{x}) = 1 - \mathbf{P}(Y = 1 \mid X = \boldsymbol{x}) = 1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}).$$

Pomocí těchto podmíněných pravděpodobností lze sestrojit algoritmus, který bude hledat parametry $\boldsymbol{w}$ takové, že budou maximalizovat správnou pravděpodobnost podle dat. Toho lze docílit sestavením věrohodnostní funkce. Pro sestavení věrohodnostní funkce využijeme faktu, že podmíněná pravděpodobnost pro obě třídy lze zapsat v kompaktním tvaru

$$\mathbf{P}(Y = y \mid X = \boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x})^y \cdot \left(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x})\right)^{(1-y)},$$

pro $y \in \{0, 1\}$. Pak dostaneme věrohodnostní funkci ve tvaru

$$L(\boldsymbol{w}) = \prod_{i-1}^{n} \mathbf{P}(Y = y_i \mid X = \boldsymbol{x}_i) = \prod_{i-1}^{n} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)^{y_i} \cdot \left(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)\right)^{(1-y_i)}.$$

Pokud na tuto funkci aplikujeme logaritmus dostaneme

$$\begin{aligned} l(\boldsymbol{w}) &= \log\left(\prod_{i-1}^{n} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)^{y_i} \cdot \left(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)\right)^{(1-y_i)}\right) \\ &= \sum_{i-1}^{n} \left(y_i \cdot \log(\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)) + (1 - y_i) \cdot \log(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i))\right) \end{aligned} \tag{3.1}$$

Nyní chceme nalézt parametry, které maximalizují věrohodnostní funkci a tedy řešíme optimalizační úlohu

$$\max_{\boldsymbol{w}} \quad l(\boldsymbol{w}).$$

Tuto maximalizační úlohu můžeme snadno řešit pomocí optimalizačních algoritmů z předchozí kapitoly. Abychom tyto algoritmy mohli použít, tak potřebujeme znát gradient a Hessovu matici účelové funkce funkce vůčí parametrům $\boldsymbol{w}$. Pro derivaci podle j-tého parametru po úpravách dostáváme

$$\frac{\partial l(\boldsymbol{w})}{\partial w_j} = \sum_{i-1}^{n} (y_i - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)) x_{i,j}.$$

Nyní můžeme snadno zapsat gradient ve tvaru

$$\nabla l(\boldsymbol{w}) = \sum_{i-1}^{n} (y_i - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)) \boldsymbol{x}_i.$$

Obdobně lze získat vztah pro Hessovu matici ve tvaru

$$\nabla^2 l(\boldsymbol{w}) = \sum_{i-1}^{n} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)) \boldsymbol{x}_i \boldsymbol{x}_i^\top.$$

Kapitola 4

Support vector machines

SVM [3] (z anglického *support vector machines*) je binární lineární klasifikátor, tzn. trénovací data pro SVM jsou ve tvaru

$$\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n},$$

kde $\boldsymbol{x} \in \mathbb{R}^d$ jsou pozorování a $y_i \in \{-1, 1\}$ je label popisující do které třídy dané pozorování patří. Cílem SVM je nalézet lineární nadrovinu oddělující uvedené dvě třídy, která je ve tvaru

$$\boldsymbol{w}^\top \boldsymbol{x} - b = 0,$$

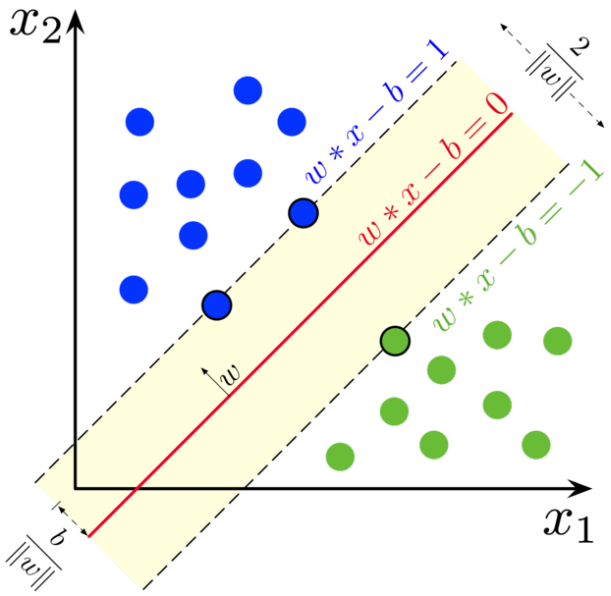
kde $\boldsymbol{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$ jsou parametry klasifikátoru a $\boldsymbol{x} \in \mathbb{R}^d$. Predpokládáme, že máme lineárně separovatelná trénovací data. V takovém případe se trénovací data dají oddělit nekonečně mnoha nadrovinami. Ale jak vybrat tu nejlepší z nich? Při hledání oddělující nadroviny se SVM snaží maximalizovat prostor okolo oddělující nadroviny ve kterém nejsou pozorování ani z jedné třídy, viz obrázek 4.1. Jinými slovy, snaží se nalézt dvě paralelní nadroviny, které oddělují uvažované dvě třídy a jejichž vzdálenost od sebe je co největší. Této vzdálenosti se říká margine a nadroviny, které maximalizují margine, vždycky procházejí trenovacími body. Tyto trénovací body se nazývají tzv. podpůrné (support) vektory. Minimální počet podpůrných vektorů, který stačí pro popsání oddělující nadroviny je dva, tzn stačí nám pouze dvě trénovací pozorování pro popsání klasifikátoru.

Formálně můžeme úlohu SVM zapsat jako následující optimalizační problém [10]

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2, \\ \text{s. t.} \quad & y_i \cdot (\boldsymbol{w}^\top \boldsymbol{x}_i - b) \geq 1, \quad i = 1, 2, \ldots, n, \end{aligned} \tag{4.1}$$

Uvedená formulace předpokládá, že data lze oddělit lineární nadrovinou dokonale. Tato formulace se nazývá SVM s tvrdým okrajem (z anglického *hard margin*). V této verzi by ovšem klasifikátor moc nefungoval. Důvod je, že nalezené oddělovací nadroviny jsou v tomto případě nalezené na základě extrémů v datech, tzn. na základě dat která jsou nejhůře rozlišitelná. Navíc tento klasifikátor vůbec neberu v úvahu data, která nejsou na okrajích.

Uvedený klasifikátor tedy může fungovat pouze v případě lineárně separiovatelných dat, což je v praxi předpoklad málokdy dosažitelný. V případě, že máme trénovací data, která nejsou



Obrázek 4.1: SVM

lineárně separovatelná, tak tuto formulaci nelze použít. V takové situaci musíme přejít z formulace (4.1) k SVM s měkkými okraji (z anglického *soft margin*), která místo tvrdého omezení na separaci do jednotlivých tříd používá relaxaci těchto podmínek

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\mu}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n}\mu_i, \\ \text{s. t.} \quad & y_i \cdot (\boldsymbol{w}^\top \boldsymbol{x}_i - b) \geq 1 - \mu_i, \quad i = 1, 2\ldots, n, \\ & \mu_i \geq 0, \quad i = 1, 2\ldots, n, \end{aligned} \tag{4.2}$$

kde $C \in \mathbb{R}$ je parametr udávající váhu na omezení a $\mu_i \in \mathbb{R}$ jsou parametry udávající penaltu do účelové funkce, pokud i-té pozorování nesplnilo omezení. Tato fomrulace tedy připouští chyby v klasifikaci a tyto chyby pouze penalizuje. Parametr C poté udává jak moc chceme tyto chyby v klasifikaci penalizovat.

Otázkou zůstává jak úlohu (4.2) řešit, protože optimalizační úlohy s omezeními jsou obecně složité na řešení. Standarní způsob jak si řešení ulehčit je použití metody Lagrangeových multiplikátorů [10]. Přepíšeme tedy úlohu ná následující tvar, kde relaxujeme omezení pomocí Lagrangeových multiplikátorů

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n}\mu_i - \sum_{i=1}^{n}\alpha_i(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i - b) - 1 + \mu_i) - \sum_{i=1}^{n}\beta_i\mu_i, \tag{4.3}$$

kde $\alpha_i, \beta_i \in \mathbb{R}$ jsou Lagrangeovy multiplikátory. Poté můžeme původní optimalizační problém přepsat na následující úlohu

$$\min_{\boldsymbol{w},b,\boldsymbol{\mu}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \quad L_p,$$

kde můžeme prohodit minimalizaci za maximalizaci a dostaneme

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \min_{\boldsymbol{w},b,\boldsymbol{\mu}} \quad L_p,$$

Pro vyřešení vnitřního minimalizačního problému spočteme derivace funkce L_p podle proměnných $\boldsymbol{w}, b, \boldsymbol{\mu}$ a položíme je rovné nule. Celkem dostáváme

$$\begin{aligned} \boldsymbol{w} &= \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i, \\ \sum_{i=1}^{n} \alpha_i y_i &= 0, \\ \sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} \beta_i &= C, \end{aligned}$$

Dosazením do rovnice (4.3) dostáváme tzv. Wolfeho duální funkci [3] ve tvaru

$$L_D = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j + \sum_{i=1}^{n} \alpha_i.$$

L_D maximalizujeme pro $0 \leqslant \alpha_i \leqslant C$, splňující navíc Karush-Kuhn-Tucker podmínky [10]:

$$\begin{aligned} \boldsymbol{w} &= \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i, \\ \sum_{i=1}^{n} \alpha_i y_i &= 0, \\ \sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} \beta_i &= C, \\ \beta_i \mu_i &= 0, \\ \alpha_i (y_i (\boldsymbol{w}^\top \boldsymbol{x}_i - b) - 1 + \mu_i &= 0, \\ y_i (\boldsymbol{w}^\top \boldsymbol{x}_i - b) - 1 + \mu_i &\geqslant 0, \end{aligned} \tag{4.4}$$

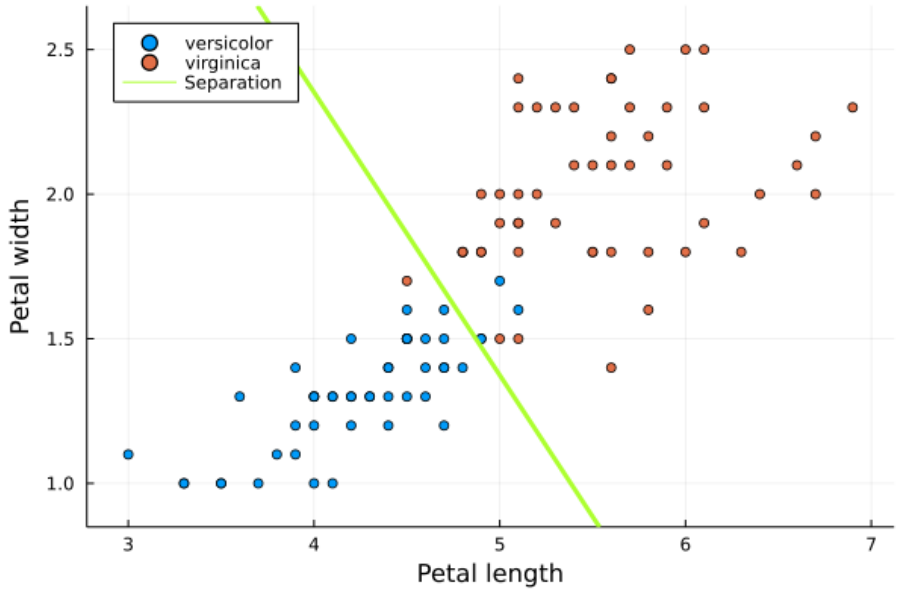
pro $i = 1, \ldots, N$. Celkem dostáváme tzv. duální formulaci [10] SVM ve tvaru

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\beta} \geq \mathbf{0}} \quad L_D, \tag{4.5}$$

4.1 Jádrový trik

Největší nevýhodou SVM zůstává to, že hledáme pouze lineární oddělovací nadrovinu. V praxi ovšem platí, že většina problému není lineárně separovatelná. Pro SVM v duálním tvaru (4.5) lze použít tzv. jádrový trik [3] (z anglického *kernel trick*), který nám umožní nalézt i nelineární separaci. Trik spočívá v tom, že nahradíme skalární součin v duální účelové funkci za skalární součin pozorování z jiného prostoru vyšší dimenze

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\beta} \geq \mathbf{0}} \quad -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i=1}^{n} \alpha_i, \tag{4.6}$$



Obrázek 4.2: Třídění dat datasetu IRIS klasifikátorem support vector machines, optimalizovaným metodou vanilla gradient descent s krokem učením , nastaveným na $\alpha = 0.001$ a omezením počtu iterací na 1000. Úspěšnost modelu je 94 procento.

kde

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^\top \Phi(\boldsymbol{x}_j),$$

kde $\Phi : \mathbb{R}^d \to \mathbb{R}^k$ je transformace do prostoru vyšší dimenze. Pro výpočet duální účelové funkce ovšem tuto transformaci nemusíme znát. Stáčí nám znát jádrovou funkci K, která počítá skalární součin v transformovaném prostoru. Typicky používané jádrové funkce jsou uvedeny v následující tabulce

Název	Vzorec
Polynomiální jádro	$K(\boldsymbol{u}, \boldsymbol{v}) = (1 + \boldsymbol{u}^\top \boldsymbol{v})^d$
Radiální jádro	$K(\boldsymbol{u}, \boldsymbol{v}) = \exp\left\{-\gamma \lVert \boldsymbol{u} - \boldsymbol{v} \rVert^2\right\}$
Gaussovské jádro	$K(\boldsymbol{u}, \boldsymbol{v}) = \exp\left\{-\frac{\lVert \boldsymbol{u} - \boldsymbol{v} \rVert^2}{2\sigma^2}\right\}$
Neuronová sít	$K(\boldsymbol{u}, \boldsymbol{v}) = \tanh(\kappa_1 \boldsymbol{u}^\top \boldsymbol{v} + \kappa_2).$

Kapitola 5

Klasifikační problém

Pro vyzkoušení klasifikačního modelu logistické regrese jsme zvolily dataset *Iris*. Tento dataset obsahuje popis tří typů kosatců pomocí šířky a délky kališního a okvětního lístku, tzn. pro každý vzorek máme celkem 5 hodnot: šířku a délku kališního lístku, šířku a délku okvětního lístku a třídu kosatce do které daný vzorek spadá. Dataset obsahuje celkem 150 vzorků. Každý vzorek datasetu je zaznamenán jako typ Float64. Jedinou výjimkou je druh kosatců, který je popsán názvem dané třídy kosatce. V datasetu nechybí žádné hodnoty.

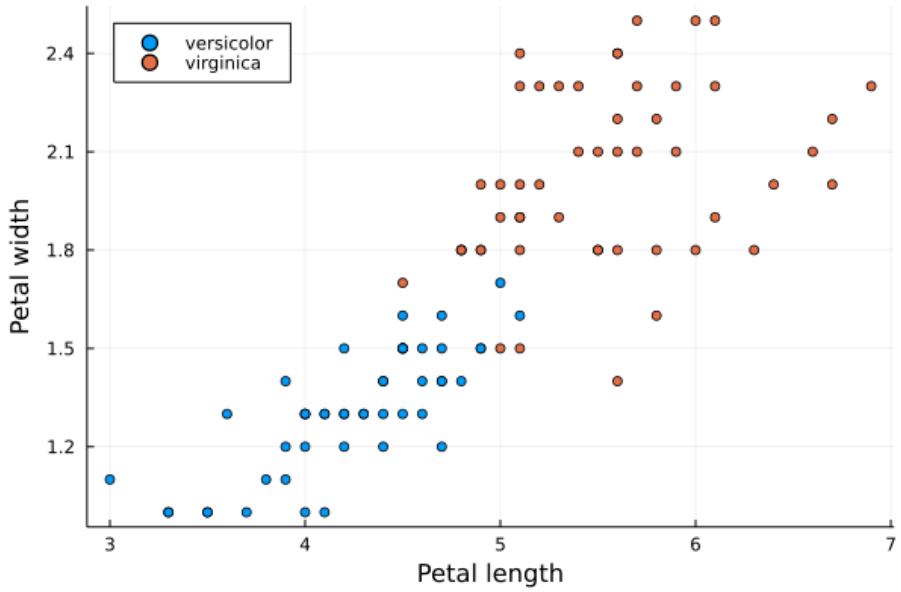
Protože jsme se v této práci zabývaly pouze klasifikačními modely pro binární problémy, tak je potřeba uvedený klasifikační problém převést na problém binární, tzn. na problém, který obsahuje pouze dvě třídy. Proto jsme z dat odstranily všechny kosatce třídy *setosa*. Dále jsme označily všechny kosatce třídy *versicolor* jako třídu negativní a přiřadily jim label $y_i = -1$. Nakonec jsme všechny kosatce třídy *virginica* jako třídu pozitivní a přiřadily jim label $y_i = 1$. Pro snažší vizualizaci a lepší představu o fungování klasifikačního modelu navíc dataset redukujeme o šířku a délku kališního lístku a tedy pro klasifikaci budeme používat pouze šířku a délku okvětního lístku. Navíc jsme každé pozorování rozšířily o 1, která bude sloužit pro natrénování interceptu. Celkem tedy dostáváme matici dat X rozměru 100×3, kde první sloupec obsahuje pouze jedničky a další dva sloupce představují šířku a délku kališního.

Pro představu jak data vypadají použijeme bodový graf, kde různé třídy odpovídají různým barvám. Graf 5.1 ukazuje závislost šířky okvětních lístků na jejich délce. Z grafu lze vypozorovat pozitivní korelaci mezi délkou a šířkou. To je přirozené, jelikož větší lístek znamená taky delší i širší lístek. Pozorujeme, že třídy jsou téměř dokonale oddělitelné a naší snahou tedy je nalézt co nejlepší oddělující nadrovinu.

Pro vyřešení modelu logistické regrese používáme gradientní metody vanilla gradient descent, momentum gradient descent, Nesterov gradient descent a metodu ADAM. Jako paramtery pro všechny optimalizační algoritmy bereme předefinované hodnoty, jako délku kroku používáme $\alpha = 0.1$ a maximální počet iterací je nastavený na hodnotu 500.

5.1 Analýza řešení

Obrázek 5.2 ukazuje výsledné oddělující nadroviny získáné pomocí jednotlivých optimalizačních algoritmů. Všechny pozorování, které se vyskytují nad oddělující nadrovinou jsou klasifikovánny jako třída kosatce *virginica*, zatímco všechny ostatní pozorování jsou klasifiko-

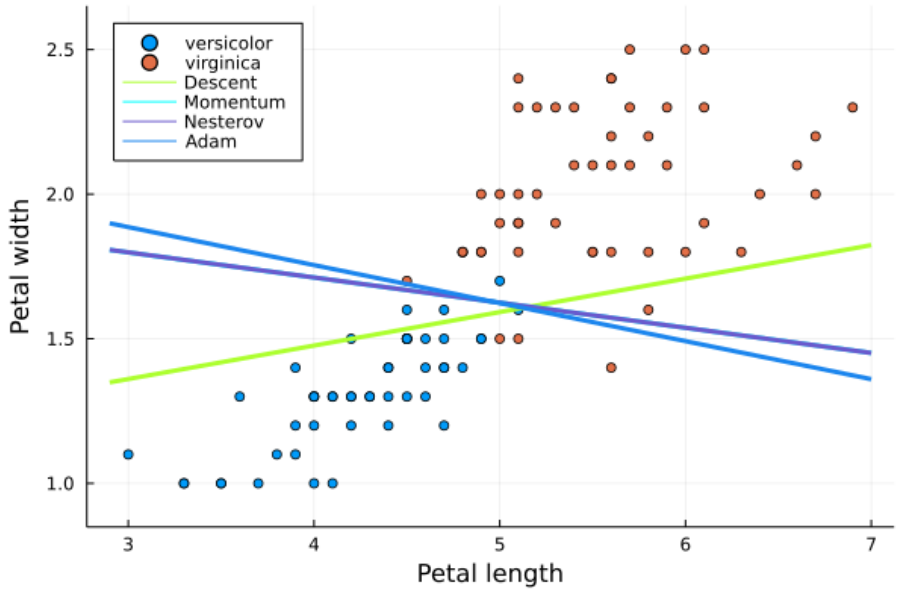


Obrázek 5.1: Vizualizace typu kosatce v závislosti na šířce a délce okvětního lístku.

vány jako třída kosatce *versicolor*. Jelikož norma gradientu pro jednotlivá řešení je rovna nule, našly jsme stacionární bod a vzhledem k tomu, že účelová funkce pro model logistické regrese je konvexní funkce, tak jsme našly globální řešení.

Obrázek 5.2 ukazuje, že existují nesprávně klasifikovaná pozorování.Přesnost řešení pro jednotlivé optimalizační algoritmy je 91% vanilla gradient descent, 95% momentum gradient descent, 95% Nesterov gradient descent a 95% pro metodu ADAM.

Vidíme že pro dataset Iris je kvalita výsledného modelu srovnatelná pro všechny optimalizační algoritmy. Z přesnosti jednotlivých modelů a z obrázku 5.2 vidíme, že metoda vanilla gradient descent měla nejnižší úspěšnost a oddělující nadrovina získána pomocí tohoto optimalizačního algoritmu se výrazně liší od ostatních. I když přesnost ostatních modelů je stejná, z obrázků pozorujeme, že lepší separující nadrovina odpovídá metodě ADAM. Podotýkáme, že řešení je hodně ovlivněno volbou počátečního bodu, hodntou kroku učení a stanoveným maximálním počtem iterací.



Obrázek 5.2: Vizualizace oddělujících nadrovin pro dataset iris nalezených pomocí různých optimalizačních algoritmů.

Závěr

Popsaly jsme klasifikační modely logistické regresi a SVM a gradientní optimalizační metody, kterými tyto modely můžeme vyřešit. Porovnaly jsme účinnost optimalizačních metod (konkrétně Vanilla gradient descent, momentum gradient descent, nesterov gradient descent, ADAM a Newton-Raphson), aplikováním na Bealovou funkci. Vyřešili jsme klasifikační problém datasetu Iris pomocí modelu logistické regrese, který jsme optimalizovaly výše zmíněnými optimalizačními metody. Porovnaly jsme získané výsledky a diskutovaly jsme vliv zvolené optimalizační metody na kvalitu výsledného modelu.

V první kapitole jsme si seznámily se zaklady teorie hledání minima funkce bez omezení, diskutovali jsme rozdíl mezi lokálním a globálním mínimem a zdůraznili jsme , že vzhledem k tomu, že zpravidla funkci můžeme vyhodnotit pouze v konečném počtu bodů, většina numerických algoritmů hledá minima lokální, nikoliv globální. Popsaly jsme také výhody konvexních funkcí, pro které platí, že jejich lokální minimum je zároveň minimem globálním. Na závěr jsme mluvily o vztahu minimalizace funkce a gradientu funkce a zdůraznily jsme, že optimalizační numerické algoritmy nehledají globální ani lokální minima, ale pouze body stacionární.

V druhé kapitole byly popsány vlastnosti, které by měla splňovat každá kvalitní optimalizační metoda. Vycházely jsmez obecného algoritmu *Descent Direction iteration*, ze kterého jsme odvodily metodu Vanilla gradient descent a drobnými modefikacemi další optimalizační algorithmy, založené na této metodě. Pro jednotlivé metody jsme popsaly iterační pravidla, podle kterých algoritmy v jednotlivých krocích (tzv. iteracích) zlepšují řešení. Uvažovaly jsme stochastickou verzi metody gradient descent, kde při výpočtu nepoužíváme celá data najednou, ale v každé iteraci pouze jejich podmnožinu.

Na příkladě Bealovou funkcí byla demonstrována účinnost optimalizačních metod Vanilla gradient descent, momentům, nesterov gradient, ADAM a Newton-Raphson. Vizualizovaly jsme a pozorovaly jak se vyvíjí řešení při hledání minima a jak se mění funkční hodnoty Bealovou funkci v těchto řešeních. Pri porovnání metod, kde za délku kroku jsme zvolily $\alpha = \alpha = 0.001$ a maximální počet iterací jsme nastavily na hodnotu 500. Ukázalo se, že metoda gradient descent je málo efektivní v případech, kdy se pohybuje delší dobu stejným směrem, jelikož neodhaduje délku kroku. Pro zlepšení tohoto chování byla vyvinuta metoda momentum gradient descent, která má ale tendenci přeskakovat minimum v důsledku naakumulované rychlostí. Nesterova metoda upravuje tuto metodu a počítá gradient v očekávaném budoucím řešení. U této metody jsme pozorovaly rychlejší pokles funkčních hodnot a trajektorii hodně podobnou trajektorii metody momentum. Metody momentum a nesterov aktualizovaly všechny parametry $\boldsymbol{\theta}$ současně a používaly stejný krok učení α. Metoda ADAM již přizpůsobuje délku kroku α_k každému parametru $\theta_1, \ldots, \theta_d$ zvlášť. Tato metoda jako jediná při daném fixním počtu iterací dokonvergovala do optimálního řešení pro Baelovu funkci a funkční hodnota klesla nejrych-

leji ze všech uvedených metod. Popsaly jsme také gradientní metodu druhého řádu Newton-Raphson, zdůraznily jsme, že její výhodou je dosažení optimálního řešení v několika iteracích a nevýhodou je citlivost na počáteční odhad řešení a relativní náročnost výpočtů, vzhledem k tomu, že pro použití potřebujeme spočítat maticí druhých derivaci (Hessovu matici). U metody Newton-Raphson trajektorie řešení byla výrazně jiná než u předchozích metod. K řešení tato metoda dokonvergovala velmi rychle (za 3 iterací). Musíme ale zdůraznit, že tyto výsledky jsou hodně ovlivněny tím, jak jsme zvolily parametr kroku učení a na jakou maximální hodnotu byl nastaven počet iteraci. Špatná volba parametrů může vést buď ke zhoršení konvergenci a tomu, že metoda nedosáhne bodu optimuma, nebo i k úplnému selhání jednotlivých metod.

Ve třetí kapitole jsme popsaly teoretické poznatky o modelu logistické regresi, který hledá separační nadrovinu pro binární klasifikační problémy. Ukázaly jsme, že cílem je maximalizovat pravděpodobnost, že vstupní prvky patří do třídy 1 a zároveň minimalizovat pravděpodobnost že patří do třídy 0.

Ve čtvrté kapitole jsme představily další lineární klasifikátor pro binární problémy zvaný SVM. Diskutovaly jsme případ lineárně separovatelných dat (nazývaný SVM s tvrdým okrajem) i případ, kdy data nejsou dokonale lineárně separovatelná (SVM s měkkým okrajem). Na závěr jsme ukázaly jádrový trik, který umožňuje použití SVM pro nalezení nelineárních oddělujících nadrovin.

V páté kapitole jsme se na příkladu datasetu Iris pokusily separovat data do jednotlivých tříd modelem logistické regresi. Pro nalezení optimálního řešení jsme použily optimalizační algoritmy Vanilla gradient descent, momentum gradient descent, nesterov gradient descent a metodu ADAM. Data jsme vizualizovaly a po nalezení rovnice pro separující nadrovinu jsme vizualizovaly i tuto nadrovinu pro všechny výše uvedené optimalizační algoritmy. Vyhodnotily jsme přesnost jednotlivých získaných modelů a ukázaly jsme, že pro náš výběr parametrů $\alpha = 0.1$ a maximální počet iterací, nastavený na 500, všechny výše uvedené metody mají vysokou úspěšnost a dobře separují data. Zároveň jsme ale pozorovaly, že metoda gradient descent špatně klasifikovala o 4 procenta pozorování více, než ostatní metody a nalezená oddělující nadrovina touto metodou se výrazně lišila od ostatních. I když metody momentum, nesterov a ADAM měly stejný počet správně klasifikovaných pozorování, na obrázku pozorovaly, že nejblíž optimální oddělující nadrovině byla nadrovina získaná metodou ADAM.

Literatura

[1] A. Ben-Israel. A newton-raphson method for the solution of systems of equations. *Journal of Mathematical analysis and applications*, 15(2):243–252, 1966.

[2] L. Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[3] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] M. Krbálek. *Funkce více proměnných (druhé přepracované vydání)*. Česká technika - nakladatelství ČVUT, 2021.

[6] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady an ussr*, volume 269, pages 543–547, 1983.

[7] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[8] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[9] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[10] L. V. Stephen Boyd. *Convex Optimalization*. Cambrige University Press, 2009.