



# SEMESTRÁLNÍ PRÁCE

## KIV-SU

### **Strojové učení**

Jakub Zíka - A15N0087P  
zikaj@students.kiv.zcu.cz

1. února 2017

# Obsah

<b>1</b>	<b>Zadání</b>	<b>4</b>
1.1	Vybrané zadání . . . . .	4
<b>2</b>	<b>Teoretický úvod</b>	<b>4</b>
2.1	Support Vector Machines . . . . .	4
2.1.1	Lineární rozhodovací hranice . . . . .	5
2.1.2	Cenová funkce lineární hranice . . . . .	6
2.1.3	Nelineární rozhodovací hranice . . . . .	7
2.1.4	Cenová funkce nelineární hranice . . . . .	7
2.2	Sequential minimal optimization (SMO) . . . . .	8
<b>3</b>	<b>Implementace</b>	<b>8</b>
3.1	Příprava dat . . . . .	8
3.1.1	Analýza a čištění . . . . .	8
3.1.2	Konverze . . . . .	9
3.2	Trénování . . . . .	10
3.2.1	Škálování a matice podobnosti . . . . .	10
3.2.2	Minimalizace cenové funkce . . . . .	10
<b>4</b>	<b>Závěr</b>	<b>11</b>
<b>5</b>	<b>Uživatelská příručka</b>	<b>11</b>
5.0.1	Požadavky . . . . .	11
5.0.2	Načtení dat . . . . .	11
5.0.3	Spuštění . . . . .	12



# 1 Zadání

Navrhnete téma zadání semestrální práce související s oblastí strojového učení. Cílem práce je prohloubit znalosti studenta v oblasti kognitivních systémů pomocí nabytých zkušeností ze semestrální práce.

## 1.1 Vybrané zadání

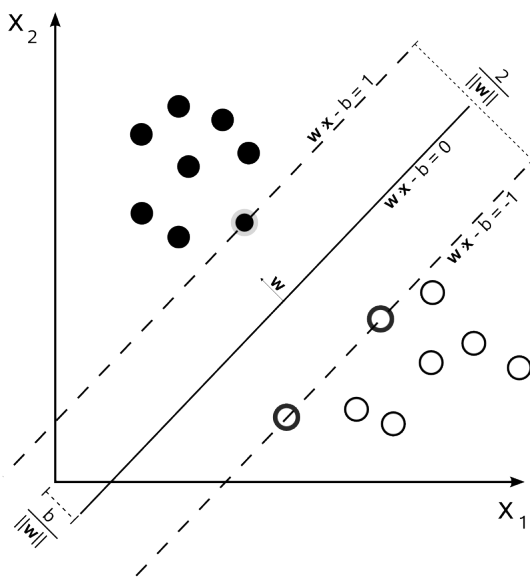
Sestrojte klasifikátor *Support Vector Machines*, dále už jen *SVM*, který bude skrze osobní údaje pasažérů lodi Titanic klasifikovat, zda daná osoba přežije či nepřežije potopení lodi. Dále se pokuste najít souvislosti mezi jednotlivými údaji o pasažérech a z nich zjistit či odvodit, které mají na přežití největší vliv. Data obsahují následující příznaky:

1. survived : přežil/nepřežil (0=nepřežil, 1=přežil)
2. pclass : socio-ekonomická třída pasažéra (1=vyšší, 2=střední, 3=nižší)
3. name : jméno
4. sex : pohlaví
5. age : věk
6. sibsp : počet sourozenců, partnerů na palubě (příbuzní stejné generace)
7. parch : počet rodičů, dětí na palubě (příbuzní odlišné generace)
8. ticket : ID lístku
9. fare : cena lístku
10. cabin : číslo kajuty
11. embarked : přístav nalodění (C=Cherbourg, Q=Queenstown, S=Southampton)

## 2 Teoretický úvod

### 2.1 Support Vector Machines

Algoritmy strojového se skládají z *trénovací množiny* a *rozhodovací hranice*. Rozhodovací hranici můžeme též nazývat *hypotéza*. Trénovací množina může obsahovat i správné odpovědi. Algoritmy tedy dělíme na *učení s učitelem* (máme



Obrázek 1: Maximální pás bez bodů trénovací množiny [?]

odpovědi) a *učení bez učitele* (nemáme odpovědi).

Každý vzorek  $\mathbf{x}$  trénovací množiny je tvořen množinou příznaků

$$x_1, x_2, \dots, x_n,$$

kde  $n$  je počet příznaků. Hypotéza má tvar  $h:x \rightarrow y$  což znamená, že hypotéza je zobrazení  $x$  do  $y$ . Hypotézu tvoří modelovací parametry

$$\Theta_n,$$

které nám umožňují nastavit rozumnou rozhodovací hranici. Jejich hodnoty předem neznáme a získáme je trénováním.

### 2.1.1 Lineární rozhodovací hranice

Metoda strojového učení, která hledá v trénovací množině umístění optimální nadroviny. Tato nadrovina slouží k rozdělení bodů projekce na dvě třídy. V tomto rozdělení je požadováno aby minimum vzdáleností bodů od této nadroviny bylo co největší. Chceme tedy, aby nadrovina měla po obou stranách co nejširší pás bez bodů. K popisu těchto pásů slouží pomocné vektory (*Support Vectors*) (viz Obr.:1). [1]

### 2.1.2 Cenová funkce lineární hranice

Metoda SVM je vylepšenou verzí *logistické regrese*. Ovšem, na rozdíl od cenové funkce logistické regrese nám SVM nevrací pravděpodobnost, ale rovnou příslušnost klasifikovaného vzorku k třídě 1 nebo 0. Zde vidíme *cenovou funkci* logistické regrese:

$$\min_{\theta} \frac{1}{m} \left[ \sum_{i=1}^m y^{(i)} (-\log h_{\theta}(x^{(i)})) + (1 - y^{(i)}) (-\log(1 - h_{\theta}(x^{(i)}))) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

Cenová funkce SVM pak vypadá následovně:

$$\min_{\theta} C \sum_{i=1}^m [y^{(i)} Cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) Cost_0(\theta^T x^{(i)})] + \frac{1}{2} \sum_{j=1}^n \theta_j^2,$$

kde *Cost* funkce pro  $y = 1$  je

$$\log \frac{1}{1 + e^{-\Theta^T x}}$$

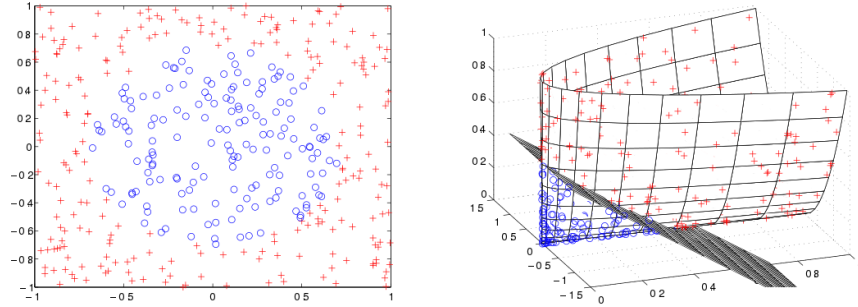
a funkce *Cost* funkce pro  $y = 0$  je

$$\log \left( 1 - \frac{1}{1 + e^{-\Theta^T x}} \right).$$

Optimalizací cenové funkce získáme hodnoty parametrů  $\Theta$ , které určují tvar hypotézy.

Hodnota  $C$  je regularizační faktor, který ovlivňuje výběr hypotézy. Rozumně vybraná hodnota pak umožní dělat ve výběru hypotézy kompromisy v extrémních případech rozdělení tříd v trénovací množině:

1.  $C$  - vysoké = malá odchylka, velký rozptyl (malá  $\lambda$ ), hrozí *overfitting*
2.  $C$  - malé = velká odchylka, malý rozptyl (velká  $\lambda$ ), hrozí *underfitting*



Obrázek 2: Ukázka lineárně neseparabilních a separabilních dat.[4]

### 2.1.3 Nelineární rozhodovací hranice

Máme-li trénovací množinu, pro kterou je rozhodovací hranice nelineární (viz. Obr.:2), musíme použít metodu jader (*Kernels*). Zavedeme si  $i$  pomocných bodů tzv. *landmarky*. Každý landmark  $l^{(i)}$  nese hodnotu třídy, do které patří. Ke každému prvku  $x^{(n)}$  trénovací množiny spočteme podobnost  $f_i$  s každým landmarkem  $l^{(i)}$ . Pro každý prvek trénovací množiny tak dostaneme vektor podobnosti  $f^{(n)}$ . Podobnost počítáme následovně:

$$f_i(x^{(n)}, l^{(i)}) = \exp\left(-\frac{\|x^{(n)} - l^{(i)}\|^2}{2\sigma^2}\right)$$

vektor podobnosti pak bude vypadat:

$$f^{(n)} = [f_1, f_2, f_3, \dots, f_i]$$

Jádrem se nazývá funkce počítání podobnosti. V tomto případě je jádro *Gaussové*. Parametr *sigma* nám určuje míru podobnosti. Máme i jiné funkce jádra jako například *Polynomiální* nebo *Lineární*. Lineární jádro je předchozí případ lineárně separabilních dat.[5],[4]

### 2.1.4 Cenová funkce nelineární hranice

Při použití gaussového jádra máme předpočítaný vektor podobnosti pro každý prvek trénovací množiny. To znamená, že vektor podobnosti může reprezentovat daný prvek trénovací množiny. Cenovou funkci tedy můžeme pozměnit do tvaru:

$$\min_{\theta} C \sum_{i=1}^m [y^{(i)} Cost_1(\theta^T f^{(i)}) + (1 - y^{(i)}) Cost_0(\theta^T f^{(i)})] + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$

Pozor, gaussové jádro je citlivé na velké rozdíly hodnot mezi jednotlivými příznaky. Je dobré příznakový vektor nejdříve naškálovat a až poté počítat cenovou funkci.

## 2.2 Sequential minimal optimization (SMO)

???

## 3 Implementace

Výsledný program je naprogramovaný v matematickém jazyce Octave. Tuto variantu jsem zvolil z důvodu velkého množství matematických operací v úloze. Jelikož výsledný program slouží pouze k vyzkoušení dané problematiky, je tento jazyk optimální pro jeho realizaci.

Program se spouští v hlavní části `main.m`. Hlavní část je pak rozdělena na `settings.m`, `preprocessing.m`, `trainManual.m`, `predictManual.m`, `statistics.m` a `evaluateSample.m`.

### 3.1 Příprava dat

Před načtením dat jsem analyzoval jednotlivé příznaky všech prvků trénovací množiny pomocí programu *Weka-3.8.1*.

#### 3.1.1 Analýza a čištění

Příprava prostředí, načtení a čištění dat probíhá v částech `settings.m` a `preprocessing.m`. Po prozkoumání jednotlivých příznaků jsem zjistil, že sloupce *embarked*, *cabin* a *age* mají některá pole nevyplněná. V příznaku *embarked* chybí 2 hodnoty, proto je jednodušší tyto řádky smazat než vymýšlet postup nahrazení. Naopak příznaku *cabin* chybí 687 z 891, což je 77%. Pokud bychom chtěli tyto mezery v datech nahradit například střední hodnotou nebo generátorem



náhodných čísel, výsledné zkreslení hypotézy by bylo příliš velké a stala by se tato operace spíše nevýhodou. Proto sloupec cabin odstraníme úplně. Posledním příznakem s chybějícími hodnotami je age. Zde chybí 177 z 891, což je 20%. To není zase tak mnoho, chybějící hodnoty nahradím středními hodnotami příznaku. Výslednou hypotézu mi to může zkreslit, ale myslím si že ne tolik, jako by se tomu stalo v případě příznaku cabin.

Příznak cabin bych odstranil i z jiného důvodu. Sice obsahuje pouze 147 hodnot, ale z toho je 101 unikátních. Pokud mám velké množství unikátních hodnot, výsledný sloupec mi poté slouží jako množina identifikátorů. Identifikátory do trénovací množiny nepřinášejí žádnou přidanou hodnotu a jsou tudíž zbytečné. Odstráním tedy i sloupec name, který je jednoznačným identifikátorem, protože má 100% unikátních hodnot.

### 3.1.2 Konverze

Jelikož nejsou všechny příznaky číselné hodnoty, je nutné je konvertovat. Patří sem sloupce age, ticket a embarked. U sloupců ticket a embarked nevím, jaké jiné hodnoty se zde mohou vyskytovat, takže převod na určitou řadu čísel zde nejde použít. Znaky každé buňky tedy převedu podle ASCII tabulky na znaky a sečtu. Náhled konverze pro sloupec embarked:

---

```
# preprocessing.m
# line 43
# EMBARKED to NUMBER
for i = 1:countRow
    data(i, 8) = sum(cell2mat (toascii(data(i, 8))));
endfor
```

---

Příznak sex lze konvertovat tak, aby jsme z toho dostaly lepší hodnotu. Jelikož víme, že pohlaví u lidí jsou pouze dvě, převedeme hodnoty male a female na binární klasifikaci, tedy hodnoty 0 a 1. Převedení vypadá následovně:

---

```
# preprocessing.m
# line 26
# Convert MALE/FEMALE to BINARY
for i = 1:countRow
    if (strcmpi(data(i, 2), 'female'))
        data(i, 2) = 1;
    else
        data(i, 2) = 0;
    endif
endfor
```

---

## 3.2 Trénování

### 3.2.1 Škálování a matice podobnosti

Před trénováním jsou data ještě škálována aby se srazili rozdíly mezi jednotlivými příznaky a došlo tak ke správnému natrénování. Příznaky se škálují odečtením maxima příznaku přes všechny vzorky trénovací množiny.

$$s_n^i = x_n^i - \max_i x_n; i \in \langle 1; m \rangle$$

Kód škálování pak vypadá následovně:

---

```
# scale.m
# line 7
for i = 1:countColumn
    for j = 1:countRow
        scaledTrainingSet(j,i) = trainingSet(j,i) / maxFeature(i);
    endfor
endfor
```

---

Před začátkem trénování vytvořím matici podobnosti  $f$ . Jako landmarky označím všechny prvky trénovací množiny. Matice podobnosti tedy bude obsahovat podobnost všech prvků trénovací množiny vůči ostatním prvkům. Matice  $f$  bude symetrická. Kód pro výpočet podobnosti je kód gaussového jádra.

---

```
# gaussianKernel.m
# line 5
function f = gaussianKernel(x1, x2, sigma)
    f = exp(-norm(x1 - x2)^2 / 2 * sigma^2);
end
```

---

### 3.2.2 Minimalizace cenové funkce

Samotné trénování probíhá v části `trainManual.m`. ???

## 4 Závěr

## 5 Uživatelská příručka

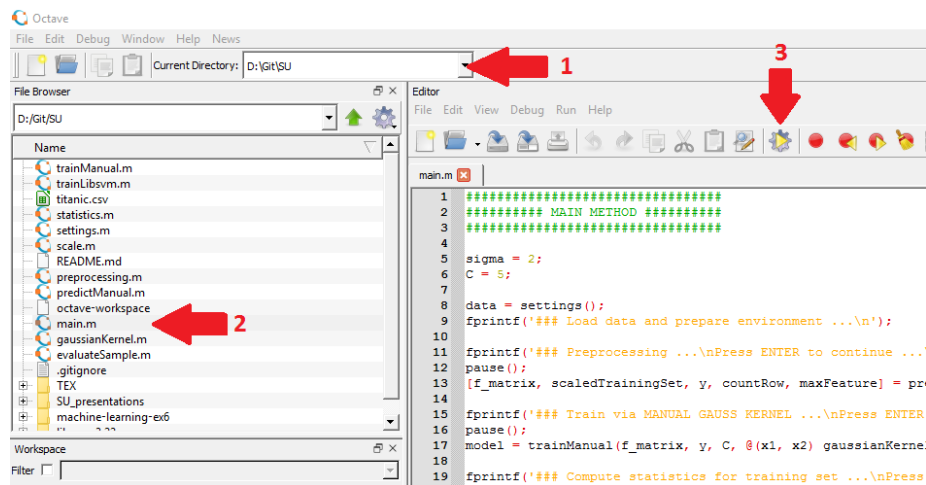
### 5.0.1 Požadavky

Aby bylo možné program spustit, je nutné mít nainstalovaný program GNU Octave ve verzi 4.2.0.

### 5.0.2 Načtení dat

Program na načítá data ze souboru `.csv`. Počítá také s tím, že data jsou ve stejné složce jako všechny potřebné moduly programu (`*.m`). Pokud jsou tedy data uloženy jinde, je nutné změnit cestu k souboru v modulu `settings.m` na řádce 25.

```
# settings.m
# line 25
data = csv2cell('titanic.csv');
```



Obrázek 3: 1. Nastavení cesty do složky programu, 2. Hlavní program (rozkliknout do záložky), 3. Spuštění programu

### 5.0.3 Spuštění

Máme dvě možnosti spuštění:

1. - Octave (GUI) : Máme-li Octave ve verzi s GUI, tak po spuštění programu navedeme Octave do správné složky s programem a spustíme modul `main.m` (viz Obr.:3)
2. - Octave (CLI) : Spouštíte-li Octave v příkazové řádce, je nutné se přesunout do složky programu s moduly a zadat pouze příkaz `main` (viz Obr.:4).

```
GNU Octave, version 4.2.0
Copyright (C) 2016 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1> cd D:\Git\SU
octave:2> main
```

Obrázek 4: Spuštění Octave (CLI)

Dále už průběh pokračuje v obou prostředí stejně. Uživatel stiskem klávesy `ENTER` načtení a úpravu dat. Jakmile jsou data připravena, spustí se výpočet matice podobnosti indikovaný hláškou `Similarity ....` Dokončení výpočtu je značeno hláškou `... Done!`. Následovně je uživatel vyzván k potvrzení trénování hypotézy. Chvilku se nic neděje, protože se matice podobnosti vektorizuje. Trénování začíná v okamžiku vyskočení hlášky `Training ...` a dokončení opět poznáme přes `... Done!`.

Program tedy přejde k výpočtu statistik a zobrazí uživatel v procentech, jak moc hypotéza natrénovaná pomocí křížové validace.

## 6 Testování

### Reference

- [1] EKŠTEIN, Kamil. *Support Vector Machines* [online]. Plzeň, 2012 [cit. 2017-01-31]. Dostupné z: <https://portal.zcu.cz/CoursewarePortlets2/DownloadDokumentu?id=123920>. Přednášky k předmětu Strojové učení. Západočeská univerzita v Plzni.
- [2] NG, Andrew. *CS229 Lecture notes - SVM* [online]. Stanford, 2016 [cit. 2017-01-31]. Dostupné z: <http://cs229.stanford.edu/notes/cs229-notes3.pdf>. Přednášky k předmětu Machine Learning - CS229. Stanford University.
- [3] NG, Andrew. *Support Vector Machines* [online]. Stanford, 2016 [cit. 2017-01-31]. Dostupné z: [https://d3c33hcgiv3.cloudfront.net/\\_246c2a4e4c249f94c895f607ea1e6407\\_Lecture12.pdf?Expires=1485993600&Signature=MGHCSj6vnfSMVWpDERGUz8fc2312duxtjEpe2o4R0vhRA9KQQuPnZOPulQy5I0mCrICpxj3M0efelTXkmFFQsKB8&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A](https://d3c33hcgiv3.cloudfront.net/_246c2a4e4c249f94c895f607ea1e6407_Lecture12.pdf?Expires=1485993600&Signature=MGHCSj6vnfSMVWpDERGUz8fc2312duxtjEpe2o4R0vhRA9KQQuPnZOPulQy5I0mCrICpxj3M0efelTXkmFFQsKB8&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A). Přednášky k předmětu Machine Learning. Stanford University.
- [4] Support vector machine. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2017-01-31]. Dostupné z: [https://cs.wikipedia.org/wiki/Support\\_vector\\_machines](https://cs.wikipedia.org/wiki/Support_vector_machines).
- [5] ZISSERMAN, Andrew. In: *SVM dual, kernels and regression* [online]. Oxford, 2015 [cit. 2017-01-31]. Dostupné z: <http://www.robots.ox.ac.uk/~az/lectures/ml/lect3.pdf>.