9. Strojové učení

B0B37NSI – Návrh systémů IoT

Stanislav Vítek

Katedra radioelektroniky
Fakulta elektrotechnická
České vysoké učení v Praze

Role strojového učení v IoT

- Prediktivní údržba
- Detekce anomálií
- Personalizace
- Monitoring prostředí
- Optimalizace využití zdrojů
- Chytrá doprava

Předpovídání ze vzorků

- Většina souborů dat jsou vzorky z nekonečné množiny.
- Nejvíce nás zajímají modely takových množin, ale máme přístup pouze k jejich vzorkům.
- Pro soubory dat sestávající z `(X,y)`
  - příznaky `X`
  - třídy (class labels) `y`
- Model je předpověď `y = f(X)`
- Model trénovaný na vzorku `D` označíme jako $f_D(X)$
- Parametrický model
  - definujme black-box popsaný množinou parametrů Θ
  - na vstupu bude příznakový vektor, na výstupu odhad třídy
  - formálně $y = f_D(X, \Theta)$
  - učení je pak určení Θ s použitím trénovací množiny

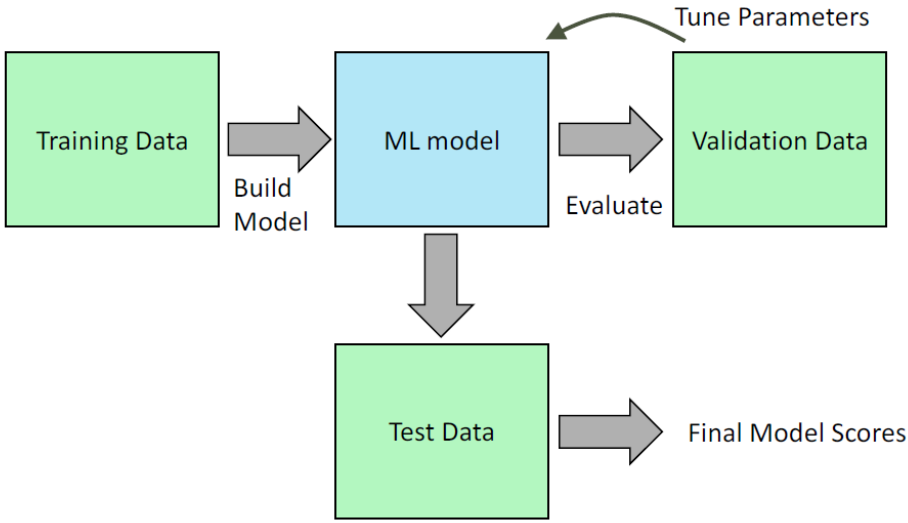
Train-Test-Validation množiny

- Při práci s ML algoritmy je třeba řešit různé problémy
- Ze vzorku dat modelujeme každý model, který se na něj hodí:
  - strukturu celé množiny
  - strukturu v konkrétním vzorku, která neplatí pro celou množinu

Příklad

- 25letý muž a 30letá žena
  - Věk dokonale předpovídá pohlaví. (věk < 27 let => muž, jinak žena)
  - Pohlaví dokonale předpovídá věk. (pohlaví == muž => 25 jinak 30)
  - Ani jeden z výsledků nelze zobecnit.
  - Tento stav nazýváme nadměrné přizpůsobení (over-fitting).

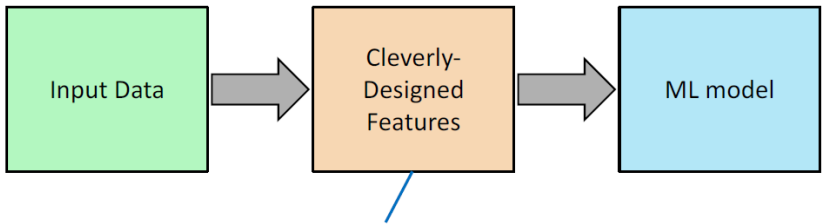
Vytváření a testování modelu



Strojovém učení

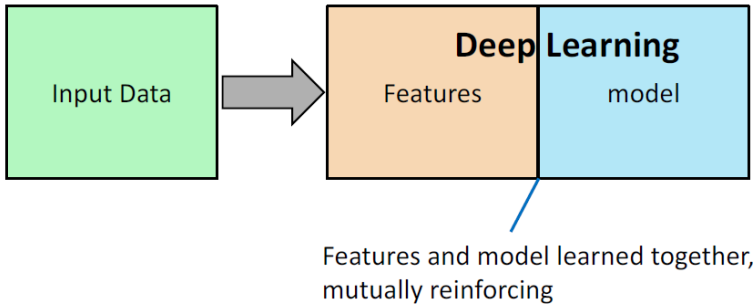
- Spousta dat, která jsou generována z mnoha zdrojů.
- Chceme
  - techniky, které minimalizují nároky na softwarové inženýrství
  - jednoduché algoritmy
  - naučit počítač učit se z dat
  - nestrávit čas psaním algoritmů nebo vysokoúrovňových funkcí
- Data bez jasných příznaků
  - data nemusí mít vždy formu tabulky, může se jednat o obrázky, videa, časové řady, dlouhé texty atp., ze kterých je těžké získat pro modely příznaky.
  - v takovém případě si musíte dát práci a nějaké příznaky z dat vydolovat (tzv. feature extraction).
  - nebo použijete algoritmy a metody, které si příznaky vytvářejí samy automaticky.
  - mezi takové metody patří umělé neuronové sítě (angl. artificial neural networks, ANN)

Stručná historie strojového učení – do roku 2012



KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012, 25. (> 106 tis. citací)

Stručná historie strojového učení – po roce 2012

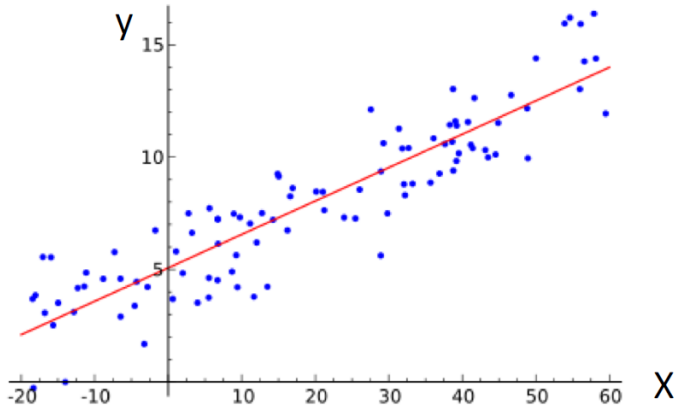


- Přístupy známé před r. 2012 se přesto stále používají
- Efektivního řešení je možné dosáhnout vhodnou volbou příznaků

Výběr správného modelu

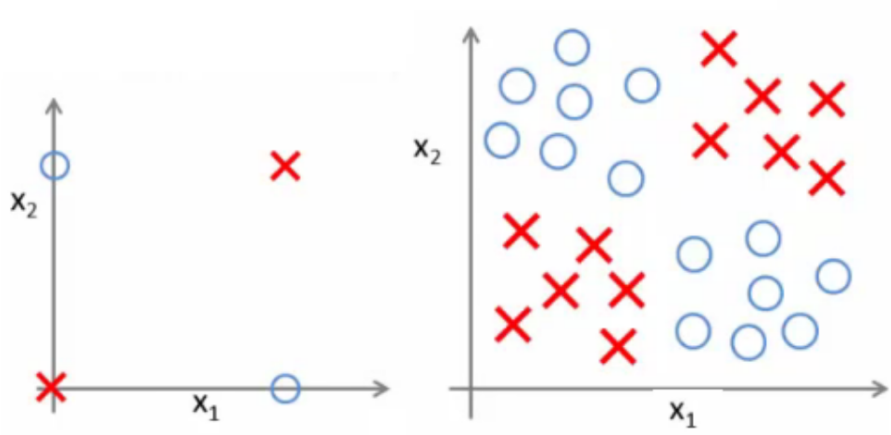
- Je třeba mít nějaké poznatky (představu) o tom, proč tento model dobře odpovídá datům nebo proč správně předpovídá třídu.
- Je třeba udržovat model jednoduchý (tj. ne příliš mnoho parametrů).
  - pak bude dostatek dat k trénování
  - trénování proběhne v rozumném čase

Predikce z dat – 1. příklad



- Lineární regrese: najdeme nejlepší přímku (lineánní funkce) modelující data

Predikce z dat – 2. příklad

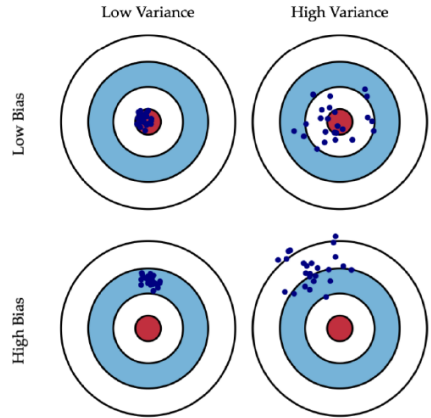


- Logistická regrese – nelineární popis (nebo později sofistikovanější metody)

Zkreslení a rozptyl

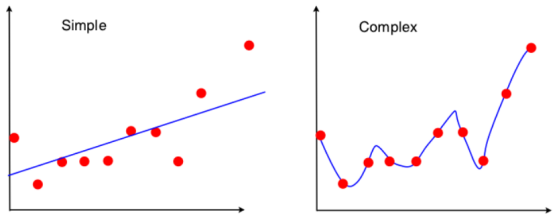
- Model generovaný z dat $f_D(X)$ je statistickým odhadem funkce $f(X)$
- Odhad je vždy zatížen zkreslením (bias) a rozptylem (variance)

- **Bias** – pokud natrénujeme model $f_D(X)$ na mnoha datasetech D, reprezentuje bias očekávanoou odchylku mezi predikcí a skutečnou hodnotou
  - $B = E[f_D(X) - y]$
  - $E[]$ se počítá přes všechny X a D
- **Rozptyl** – pokud natrénujeme model $f_D(X)$ na mnoha datasetech D, reprezentuje rozptyl V rozptyly jednotlivých odhadů
  - $V = E[(f_D(X) - \bar{f}(X))^2]$
  - kde $\bar{f}(X)$ je průměrná predikce X



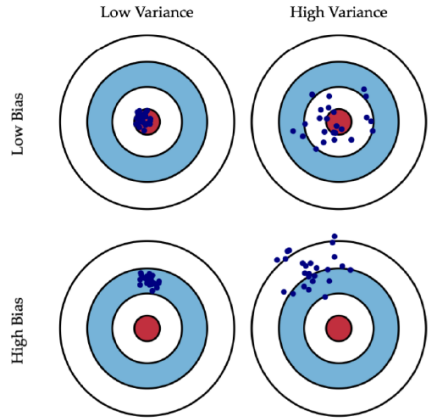
Kompromis mezi zkreslením a rozptylem 1/2

- Obvykle dochází ke kompromisu mezi zkreslením a odchylkou, který je způsoben složitostí modelu.
  - Složité modely (mnoho parametrů) mají obvykle nižší zkreslení, ale vyšší rozptyl.
  - Jednoduché modely (málo parametrů) mají vyšší bias, ale nižší rozptyl.
- Příklad:
  - Lineární model může odpovídat pouze přímce, zatímco polynom vysokého stupně může odpovídat složité křivce.
  - Polynom může odpovídat spíše jednotlivým vzorkům než celé množině, jeho tvar se může lišit vzorek od vzorku, takže má vysoký rozptyl.



Kompromis mezi zkreslením a rozptylem 2/2

- Celková očekávaná chyba je $B^2 + V$
- Je třeba najít kompromis mezi příspěvky
- Pokud dominuje rozptyl, znamená to příliš velký rozptyl mezi modely
  - over-fitting
  - model má špatnou prediktivní schopnost. protože příliš reaguje na drobné výkyvy v tréninkovém modelu. dat
- Pokud dominuje odchylka, nemodeluje model data dost dobře
  - under-fitting
  - může nastat např. pokud použijeme lineární model na nelineární data



Rozdělení metod strojové učení

- **supervised learning** – učení na základě označeného příkladu, s učitelem
  - učitelem jsou známé hodnoty veličiny, kterou se snažíme na základě modelu predikovat, resp. pochopit, na čem závisí
  - např. detektor e-mailového spamu
  - úžasně efektivní, pokud máte k dispozici spoustu příkladů
- **unsupervised learning** – objevování vzorů, bez učitele
  - nemáme žádnou veličinu a snažíme se prostě v datech vyznat
  - např. shluková analýza
  - v praxi obtížné, ale užitečné, pokud vám chybí označené příklady.
- **reinforcement learning** – zpětná vazba správně/špatně, posilování modelu
  - Např. učení se hrát šachy na základě vítězství nebo prohry.
  - funguje dobře v některých oblastech, nabývá na významu

Strojové učení s učitelem

- Predikce ceny nemovitosti s ohledem na velikost
  - Cena je spojitou funkcí velikosti, takže se jedná o problém regrese
- Další příklady
  - Je na tomto obrázku kočka, pes, auto, dům?
  - Jak by tento uživatel ohodnotil tuto restauraci?
  - Je tento e-mail spam?
  - Je tato skvrna supernova?

Strojové učení bez učitele

- Rozdělte ručně psané číslice do 10 tříd.
- Vezměte kolekci 1000 esejí napsaných o ekonomice USA a najděte způsob, jak automaticky seskupit tyto eseje do malého počtu, které jsou si nějakým způsobem podobné nebo spolu souvisejí hodnotou proměnných, jako je četnost slov, délka vět, počet stran atd.
- Jakých 20 témat je právě teď na Twitteru nejčastějších?
- Vyhledání a seskupení odlišných přízvuků lidí v dané lokalitě
- Do nesupervizovaného učení také (obvykle) spadá i detekce anomálií (angl. anomaly detection) – např. banka se snaží najít podezřelé transakce (fraud detection, ochrana proti zneužití karty, atp.)
- Dalším příkladem problému řešeného pomocí zkoumání dat je tzv. doporučování (angl. recommendation) – vlastníte-li e-shop (příp. internetový časopis, iTunes, Netflix atp.), snažíte se na základě dat o zákaznících a zejména zákazníkovi, který právě prohlíží Vaše stránky, odhadnout, co by si tak mohl ještě chtít koupit (přečíst, podívat, poslechnout) a to mu ukázat.

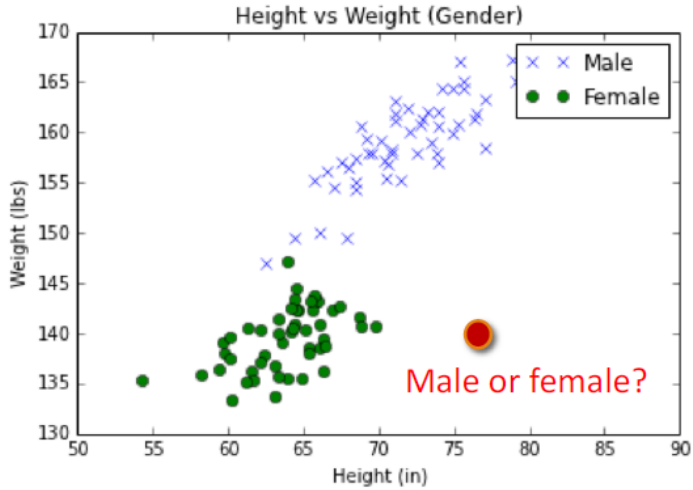
Klasifikace

- Supervizované učení: Snažíme se zjistit, jak vysvětlovanou proměnnou Y ovlivňují příznaky $X_0, X_1, ..., X_{p-1}$, hledáme tedy nějaký funkční vztah tak, aby *co nejvíce platilo*

$$Y f(X_0, X_1, ..., X_{p-1})$$

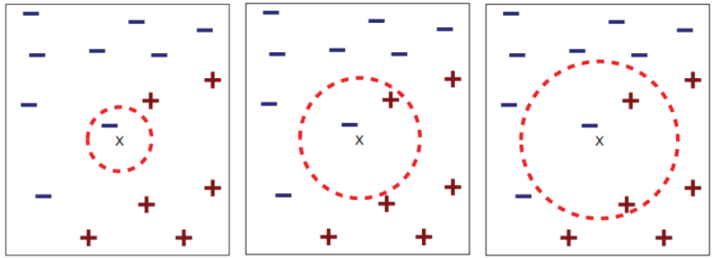
- Funkce f nemusí být nutně podobná funkcím, které znáte z analýzy
- Tvar hledané funkce často ovlivňuje to, jakých hodnot může nabývat vysvětlovaná proměnná Y
  - Může-li nabývat jen několik málo hodnot, mluvíme o problému **klasifikace** (angl. classification). Sem spadá např. určení, jestli pacient má/nemá nemoc, jaké písmeno je (ručně) napsáno na obrázku, atp.
  - Může-li nabývat tolika hodnot, že je rozumnější ji považovat za spojitou, mluvíme o problému **regrese** (angl. regression).
- Populární metody
  - Nearest neighbor
  - Decision tree
  - Support vector machine

Klasifikace



k-NN – k-nejbližších sousedů

- Ve fázi učení se předzpracuje trénovací množina tak, aby všechny příznaky měly střední hodnotu 0 a rozptyl 1 – toto umístí každý prvek trénovací množiny do některého místa v N-rozměrném prostoru.
- Ve fázi klasifikace umístím dotazovaný prvek do téhož prostoru a najdu k nejbližších sousedů. Objekt je pak klasifikován do té třídy, kam patří většina z těchto nejbližších sousedů.



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

k-NN metriky

- Euclidovská vzdálenost – nejjednodušší, jednoduchá na výpočet

$$d(x, y) = \|x + y\|$$

- Kosinové vzdálenost – obrázky, dokumenty, ...

$$d(x, y) = 1 - \frac{x \cdot y}{\|x\| \|y\|}$$

- Jaccardova vzdálenost – množiny
- Hammingova vzdálenos – řetězce

$$d(x, y) = \sum_{i=1}^{N} ()x_i \neq y_i)$$

- Manhattan – souřadnice

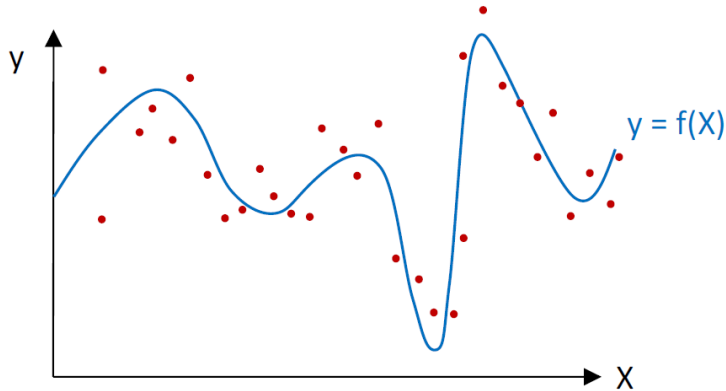
$$d(x, y) = \sum_{i=1}^{N} |x_i - y_i|$$

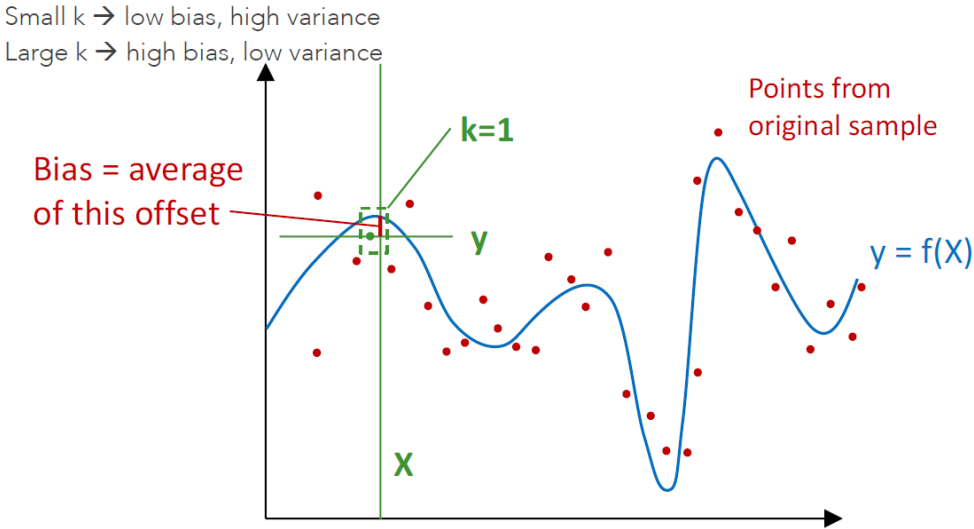
Volba parametru k 1/6

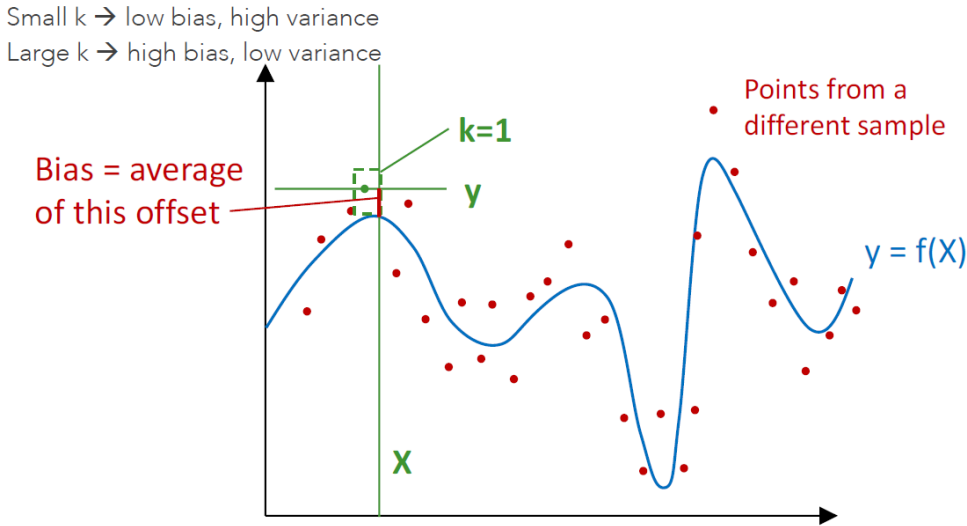
- Připomeňme, že chyby předpovědi lze rozdělit do dvou hlavních kategorií:
  - Chyby způsobené zkreslením – rozdíl mezi očekávanou (nebo průměrnou) předpovědí našeho modelu a správnou hodnotou, kterou se snažíme předpovědět
  - Chyby způsobené rozptylem – variabilita předpovědi pro daná vstupní data
- Existuje kompromis mezi schopností modelu minimalizovat zkreslení a rozptyl.
- V závislosti na hodnotě k existuje kompromis mezi zkreslením a rozptylem:
  - Malé $k \rightarrow$ nízké zkreslení, vysoký rozptyl
  - Velké $k \rightarrow$ vysoké zkreslené, nízký rozptyl

Volba parametru k 2/6

- Předpokládejme, reálná data jsou popsána modrou křivkou
- Systém je zatížen aditivnim šumem s nulovou střední hodnotou
- Červené body představují vzorky signálu

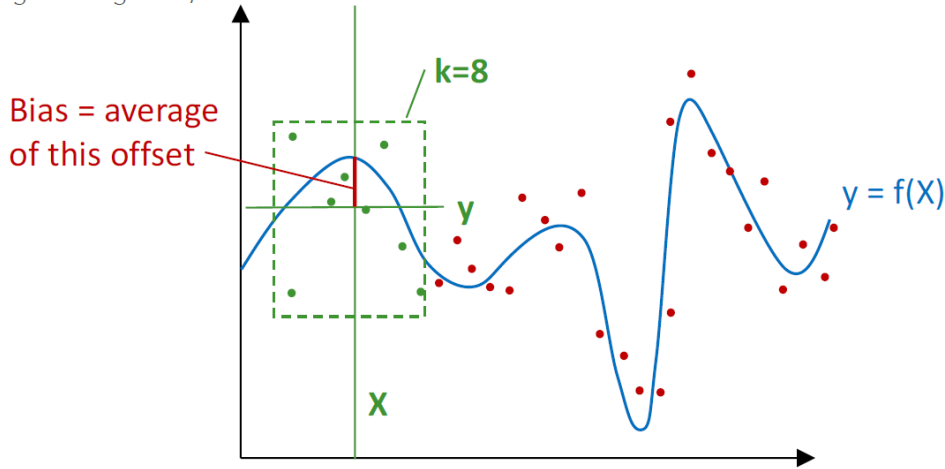


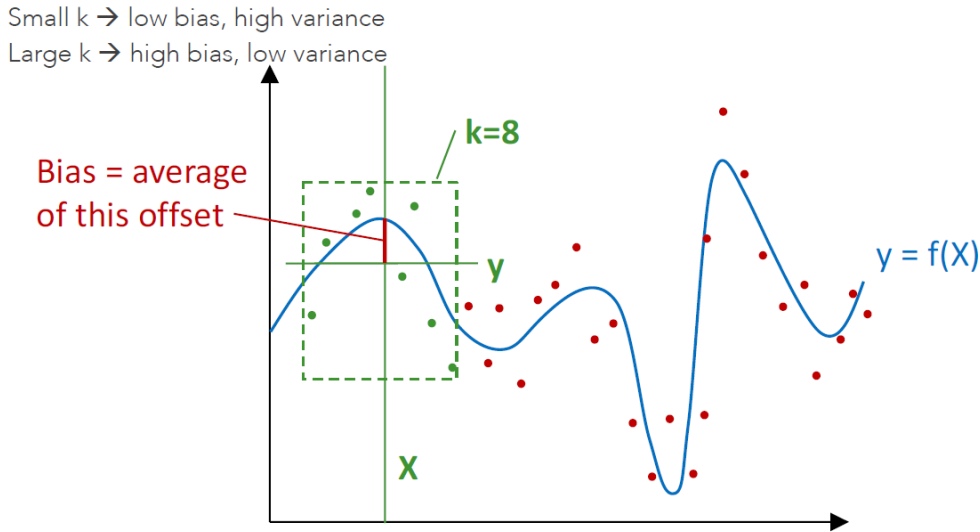

Small k → low bias, high variance
Large k → high bias, low variance
Points from original sample
k=1
Bias = average of this offset
y
y = f(X)
X


Small k → low bias, high variance
Large k → high bias, low variance
Bias = average of this offset
k=1
y
X
Points from a different sample
y = f(X)

Volba parametru *k*

Small k → low bias, high variance
Large k → high bias, low variance




Small k → low bias, high variance
Large k → high bias, low variance
k=8
Bias = average of this offset
y
y = f(X)
X

Volba parametru k – praktické rady

- Použijte křížové ověřování
  - Rozdělte data na trénovací, validační a testovací podmnožiny, např. 60-20-20% náhodné rozdělení.
- Předpovídejte
  - Pro každý bod ve validační množině proveďte predikci pomocí k-nejbližších sousedů z trénovací množiny.
  - Změřte chybovost (klasifikace) nebo kvadratickou chybu (regrese).
- Vylaďte
  - Vyzkoušejte různé hodnoty k a použijte tu, která dává minimální chybu na validační množině.
- Vyhodnoťte
  - Otestujte na testovací množině a změřte výkonnost

- Kdy vybrat metodu k-NN?
  - Hodně trénovacích dat
  - Malá množina příznaků (do 20)

k-NN – příklad 1/3

- Dataset: informace o lidech, kteří nakoupili na základě reklamy
- https://www.kaggle.com/datasets/rakeshrau/social-network-ads

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
```

- Načtení dat

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [1, 2, 3]].values
y = dataset.iloc[:, -1].values
```

- Rozdělení dat na trénovací (80%) a testovací množinu

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train, y_test = train_test_split(X, y, test_size =
    0.20, random_state = 0)
```

- Škálování – práce s menšími čísly

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

```

- Natrénování modelu

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = '
    minkowski', p = 2)
classifier.fit(X_train, y_train)
```

- Při vytváření modelu používáme 3 parametry. `n_neighbors` je nastaveno na 5, což znamená, že pro klasifikaci daného bodu je potřeba 5 bodů okolí.

k-NN – příklad

- Je použita Minkowského metrika

$$\left(\sum_{i=1}^{N} |x_i - y_i|^p \right)^{1/p}$$

- Hodnota p
  - p = 1, Manhattan Distance
  - p = 2, Euclidean Distance (takže lze nastavit metric na euclidean)
  - p = infinity, Cheybchev Distance

```
y_pred = classifier.predict(X_test)
```

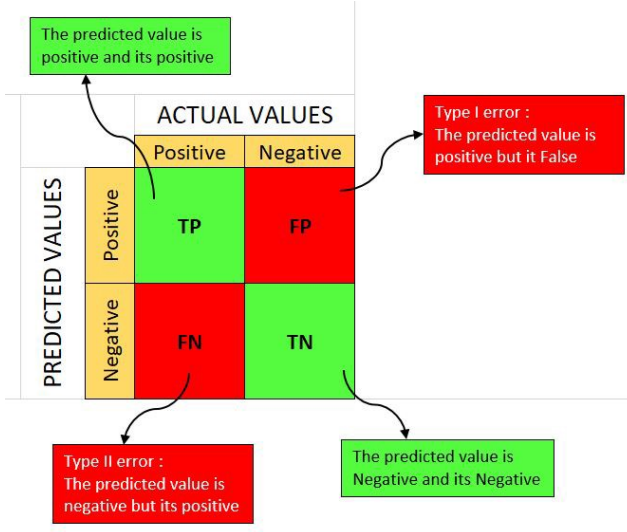
- Vyhodnocení

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test,y_pred)
```

Precission a Recall, Confusion Matrix 1/2

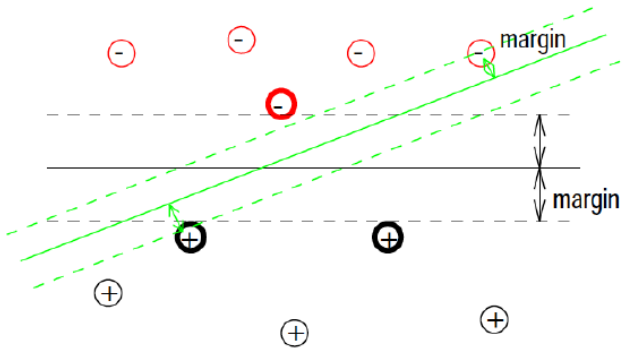
- Oba pojmy – precision i recall – jsou původně převzaty z informatiky a je možné je přeložit jako přesnost a výtěžnost pokrytí zkoumaných dat daným přístupem.
- Míra precision je definována jako procentuální poměr relevantních výsledků analýzy ke všem výsledkům analýzou získaným.
- Na rozdíl od precision je recall poměr relevantních výsledků analýzy ke všem relevantním výskytům ve zkoumaném vzorku bez ohledu na to, zda byly analýzou identifikovány.
- Celkový objem dat, s nimiž pracujeme při analýze, můžeme rozdělit do čtyř skupin:
  - případy, které nás zajímají a pomocí dané metody se nám skutečně podařilo je vyfiltrovat (relevantní výsledky; angl. tzv. true positives, TP - správná zařazení do výsledků)
  - případy, které nás sice nezajímají, ale naše metoda (jsouc nedokonalá) je vyfiltrovala taky (false positives, FP - nesprávná zařazení)
  - případy, které nás sice zajímají, ale naší metodě unikly (false negatives, FN)
  - případy, které nás nezajímají a metoda je z výsledků správně vyloučila (true negatives, TN)
- precision $= TP / (TP + FP)$, recall $= TP / (TP + FN)$

Precission a Recall, Confusion Matrix 2/2



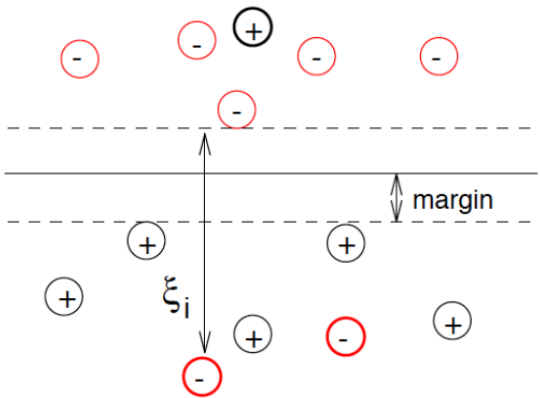
Support Vector Machine

- Nalezení lineární rozhodovací hranice (roviny), která má nejmenší chybu zobecnění
- Nejlepší rovina by měla mít největší rozpětí (margin)



Co když řada dat není lineárně separabilní?

- SVM zahrnuje penalizaci za nesprávnou klasifikaci



Regularization parameter to weight training error

$$\min \; ||w||^2 + C \sum_{i=1}^{m} \xi_i$$

1/Margin

Classification error in training set

Support Vector Machine – příklad 1/2

- Použijeme dataset s 30 příznaky získaných analýzou snímku s nádorem prsu
- Nádor je buď zhoubný (malignant) nebo nezhoubný (benign)

```
from sklearn import datasets
cancer = datasets.load_breast_cancer()

print("Features: ", cancer.feature_names)
print("Labels: ", cancer.target_names)
```

- Rozdělení datasetu na trénovací (70%) a testovací množinu

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(cancer.data,
    cancer.target, test_size=0.3, random_state=109)
```

Support Vector Machine – příklad 2/2

- Trénování modelu

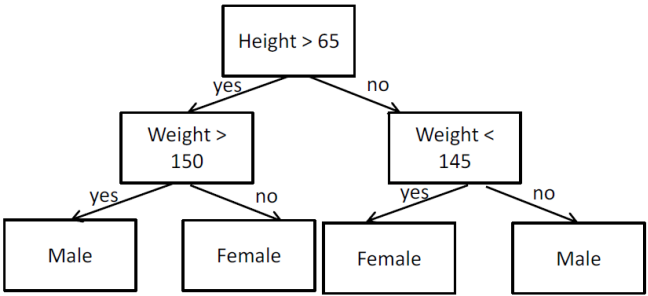
```
from sklearn import svm

clf = svm.SVC(kernel='linear') # Linear Kernel
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

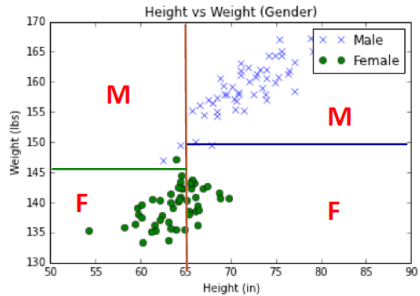
- Evaluace modelu

```
from sklearn import metrics

# Accuracy:  how often is the classifier correct?
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

# Precision:  percentage of positive tuples are labeled as such?
print("Precision:", metrics.precision_score(y_test, y_pred))

# Recall:  what percentage of positive tuples are labelled as such?
print("Recall:", metrics.recall_score(y_test, y_pred))
```

Rozhodovací stromy 1/3

- Motion tracking (Xbox Kinect)
- Stromová struktura, kde uzly jsou "otázky" a listy určují klasifikaci nebo označení (label)





Rozhodovací stromy 2/3

- Obecně neparametrické, není třeba definovat hloubku nebo počet uzlů.
- Cílem učícího algoritmu je najít příznaky, které poskytují největší diskriminaci (rozdělení) na základě třídy.
- Například pro určení pohlaví poskytne dotaz na něčí výšku poměrně nízkou chybovost, kdežto něco jako barva očí poskytuje málo informací.
- Algoritmus:
  - Najdi příznak, který poskytuje nejlepší přesnost separace
  - Vytvoř uzel a odděl data podle této vlastnosti
  - Pokud je chyba nulová (v příkladu všechny vzorky s výškou > 72 jsou muži, vytvoř list – klasifikaci)
  - V opačném případě rekurze pro každou vytvořenou podmnožinu

Rozhodovací stromy 3/3

- Výhody:
  - Model je velmi snadno prohledatelný, můžete se podívat na každý uzel a snadno pochopit jeho účel (tj. funkci a práh nebo hodnotu).
  - Dokáže pracovat s heterogenním daty (reálná čísla, kategorie, všechna dohromady)
  - Náklady na klasifikaci jsou $O(logn)$, kde n hloubka stromu
- Nevýhody:
  - Optimalizace je s největší pravděpodobností plná lokálních optim a jedná se o NP-úplný problém (tj. přesný výpočet je příliš drahý).
  - Velká citlivost na rozložení trénovacích dat (časté přetrénování!).

Příklad rozhodovacího stromu

- Velmi často je klasifikační problém **binární**, kdy proměnná Y může mít jen dvě hodnoty.
- My si použití stromu ukážeme na (vymyšlených) datech a problému určování, jestli pacient má či nemá závažnou nemoc známou jako *rýmička*.
- Příznaky budou pro jednoduchost také binární: Pohlaví (žena/muž), horečka ($> 39°C/\leq 39°C$) a to, jestli daný člověk zvládl/nezvládl vstát z postele.
- Ukážeme si dva rozhodovací stromy a porovnáme si, jak je který z nich dobrým modelem následujících dat:

rýmička	pohlaví	$> 39°C$	vstal(a)?
ano	muž	ne	ne
ne	žena	ano	ano
ne	muž	ne	ano
ano	žena	ano	ne

Logistická regrese

- Logistická regrese je pravděpodobně nejpoužívanějším klasifikátorem pro obecné účely.
- Je velmi dobře škálovatelný a jeho trénování je velmi rychlé.
- Používá se pro
  - filtrování nevyžádané pošty
  - klasifikace zpravodajských zpráv
  - klasifikace webových stránek
  - klasifikace produktů
  - většina klasifikačních problémů s velkými a řídkými soubory příznaků.
- Nevýhoda: přeučení v případě velmi řídkých dat → často se používá s regularizací
- Je to binární klasifikátor, popsaný logistickou funkcí

$$P(X) = \frac{1}{1 + exp(-X\beta)}$$

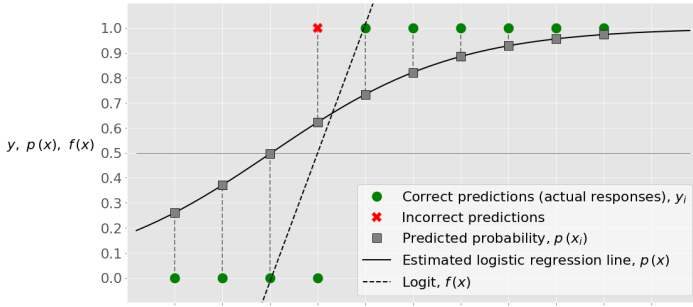
- Výhody:
  - Rychlý (trénování i klasifikace) a jednoduchý model
  - Výsledek klasifikace má jednoduché vysvětlení

Logistická regrese – příklad 1/3

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

x = np.arange(10).reshape(-1, 1)        # vstupni data
y = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])  # vystupni data

# model
model=LogisticRegression(solver='liblinear',random_state=0).fit(x,y)
# solver:  liblinear, newton-cg, lbfgs, sag, saga
# fine tunning:  regularizace - C = 10.0
# tridy modelu:
print(model.classes_)
# vlastnostii reg.  krivky:
print(model.intercept_)
print(model.coef_)
```

Logistická regrese – příklad


```
# pravdepodobnosti
model.predict_proba(x)
# predikovany vystup
model.predict(x)
# pocet spravnych
model.score (x, y)
```



Logistická regrese – příklad 3/3

```
cm = confusion_matrix(y, model.predict(x))

fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'
    ))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
  for j in range(2):
    ax.text(j, i, cm[i, j], ha='center', va='center', color='red')
plt.show()

# report
print(classification_report(y, model.predict(x)))
```

Porovnání klasifikátorů

	Decision Tree	Logistic Regression	SVM
Model	Tree (horizontal/vertical separating plane)	Linear separating plane	Can also use non-linear separating plane
Simplicity	Very simple	Simple	Complicated
Interpretation of output	Clear, follow the questions asked	Probability	Unclear
Chances of over-fit	High	Less worry	Should be careful
Development time	Just plug data in	Just plug data in	Tuning tuning tuning
Off-the-shelf accuracy	Okay	Okay	Probably the highest
Use case	When features are clear	Often combined with neural networks	Often suggest RBF as a starting point if you have zero knowledge on data

Random Forests

- Zástupce skupiny **ensamble** method (další třeba AdaBoost)
  - namísto jednoho modelu (např. rozhodovacího stromu) použijeme více modelů a jejich predikce nějakým způsobem zkombinujeme do finálního rozhodnutí.
- Pro jednoduchost předpokládejme, že máme binární klasifikační problém, tj. rozhodujeme jestli $Y = 0$ nebo $Y = 1$
  1. Ze vstupního trénovacího datasetu D vytvoříme n datasetů $D_1, ..., D_n$ stejně velkých jako D pomocí metody **bootstrap**, neboli pomocí výběru s opakováním.
  2. Na každém datasetu D_i naučíme rozhodovací strom, označme tyto stromy $T_1, ..., T_n$
  3. Každý datový bod (tj. řádek z tabulky s daty D) proženeme všemi stromy $T_1, ..., T_n$ a od každého z nich si uložíme rozhodnutí $Y_1, ..., Y_n$
  4. Všechny tyto stromy $T_1, ..., T_n$ tvoří náhodný les a jeho finální rozhodnutí o hodnotě Y je dané většinovým rozhodnutím stromů, je-li např. v množině $Y_1, ..., Y_n$ více jedniček než nul, je predikce náhodného lesa $Y = 1$

Bootstrap

Ukážeme si, jak funguje bootstrap na jednoduchém příkladu a našem datasetu:

id	rýmička	pohlaví	> 39°C	vstal(a)?	věk
1	ano	muž	ne	ne	65
2	ne	žena	ano	ano	34
3	ne	muž	ne	ano	72
4	ano	žena	ano	ne	20
5	ano	muž	ne	ano	45

Chceme-li vytvořit "bootstrapem" dataset velikosti pět, pětkrát si náhodně vybereme řádek s tabulky s tím, že se řádky v našem výběru mohou opakovat. Vybereme-li např. řádky s id 1,4,3,3,1, dostaneme dataset

id	rýmička	pohlaví	> 39°C	vstal(a)?	věk
1	ano	muž	ne	ne	65
2	ne	žena	ano	ano	34
3	ne	muž	ne	ano	72
1	ano	muž	ne	ne	65
3	ne	muž	ne	ano	72

Random Forest – příklad 1/3

```
from sklearn import datasets
iris = datasets.load_iris()
# print the label species(setosa, versicolor,virginica)
print(iris.target_names)
# print the names of the four features
print(iris.feature_names)
# print the iris labels (0:setosa, 1:versicolor, 2:virginica)
print(iris.target)
# Creating a DataFrame of given iris dataset.
import pandas as pd
data=pd.DataFrame({
  'sepal length':iris.data[:,0], 'sepal width':iris.data[:,1],
  'petal length':iris.data[:,2], 'petal width':iris.data[:,3],
  'species':iris.target
})
```

Random Forest – příklad 2/3

```
from sklearn.model_selection import train_test_split

# Features
X=data[['sepal length','sepal width','petal length','petal width']]

# Labels
y=data['species']

# Split dataset into training set (70%) and test (30%)set
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)

#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
```

Random Forest – příklad 3/3

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

# Manual prediction
clf.predict([[3, 5, 4, 2]])
```