Katedra geomatiky

Strojové učení v dálkovém průzkumu Země

Lekce 8: Sborové učení (ansámbly)
[bootstrap aggregating, stohování, zesilování]

Přednášející: Lukáš Brodský lukas.brodsky@natur.cuni.cz, Ondřej Pešek ondrej.pesek@fsv.cvut.cz a Martin Landa martin.landa@fsv.cvut.cz

Položte záludnou otázku:

1. zeptejte se tisíce lidí;
2. slučte jejich odpovědi;
3. je odpověď odborníka lepší než odpověď davu?
   - diskuse viz Platón: Lachés, 184e

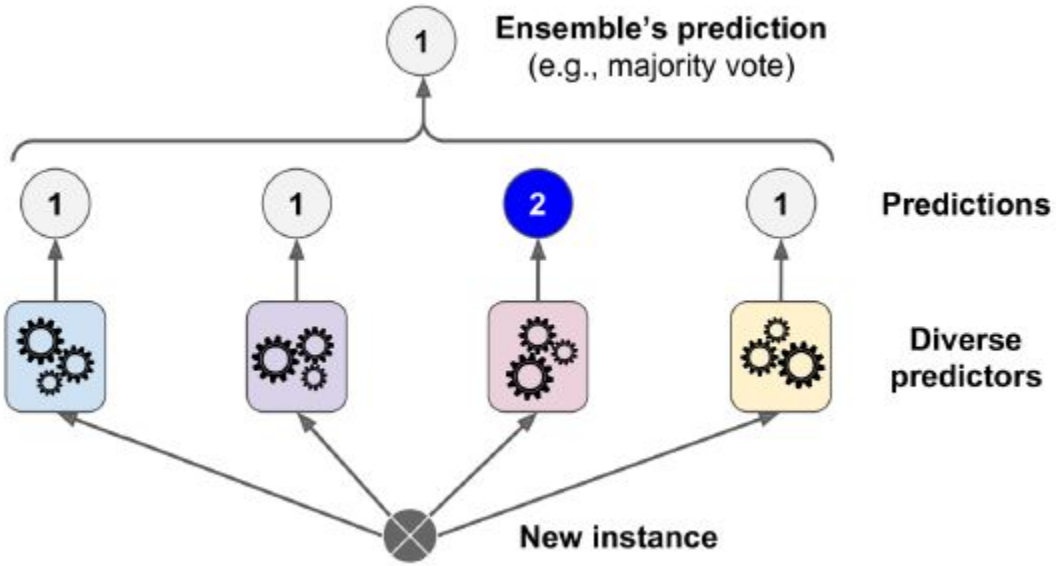
Sbor odhadujících = ansámbl (*ensemble*)

Hlasovací klasifikátor (voting classifier)

- 1/ Natrénujte několik klasifikátorů, každý dosahující přesnosti 80 %
- 2/ Slučte klasifikace a zvolte tu třídu, která
  - získá nejvíce hlasů – klasifikátor pevné volby (hard voting classifier)
  - je předpovězena s největší průměrovanou pravděpodobností – měkké hlasování (soft voting classifier)

- Hlasovací klasifikátor dosáhne vždy lepších výsledků než nejlepší z klasifikátorů ve sboru, a to i pokud jsou ve sboru klasifikátory nedostatečné kvality
  - Za předpokladu dostatečné rozmanitosti klasifikátorů ve sboru
  - Za předpokladu nezávislých, nekorelovaných klasifikátorů
  - Podmínek lze dosáhnouti trénováním na odlišných datech

Hlasovací klasifikátor



nekorelované!

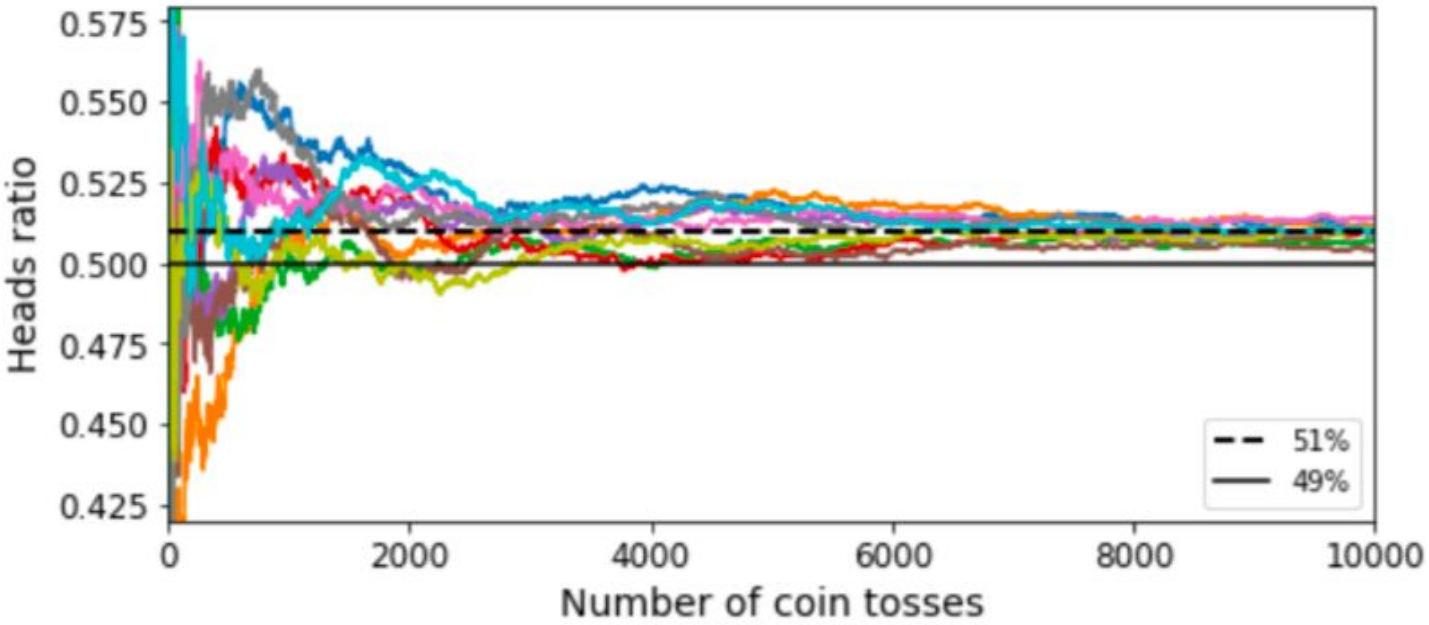
Proč?

Analogie

- Lehce *cinknutá* mince: 51% šance hlavy
- 1 000 hodů: 510 / 490
  - pravděpodobnost: 75 %
- 10 000 hodů:
  - pravděpodobnost: 97 %

Zákon velkých čísel

Zákon velkých čísel



Hlasovací klasifikátor

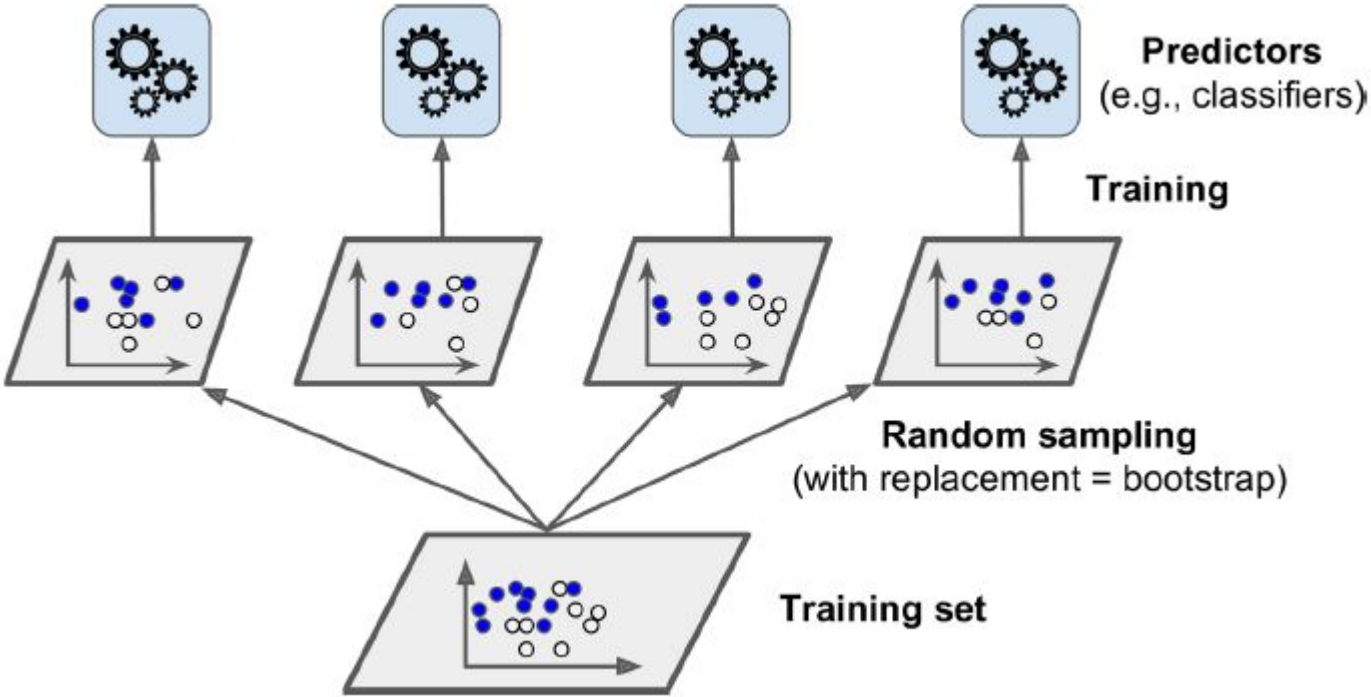
- Sklearn API:

```
from sklearn.ensemble import  VotingClassifier

voting_clf = VotingClassifier(
      estimators=[('lr, log_clf), ('svm', svc_clf), (...), … ],
      voting='hard'
)

voting_clf.fit(X_train, y_train)
```

Bagging a pasting

Bagging a pasting

- Bagging (bootstrap aggregating)
  - vzorkování s opakováním
- Pasting
  - vzorkování bez opakování


Predictors
(e.g., classifiers)
Training
Random sampling
(with replacement = bootstrap)
Training set

Bagging a pasting – Slučování

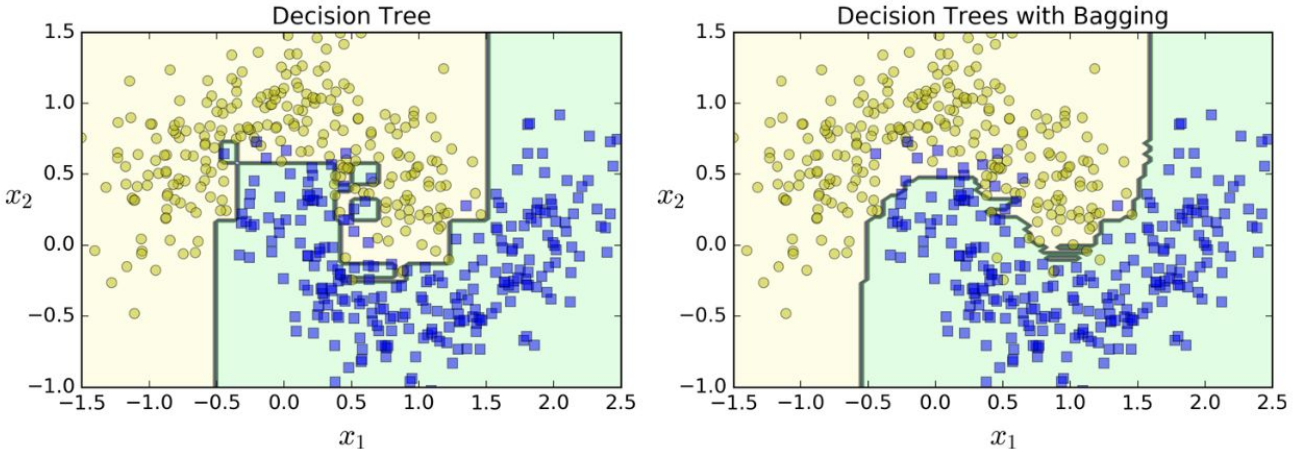
- Statistický model
- Průměrová regrese

Bagging a Pasting

- Každý klasifikátor je zaujatější (větší bias) než kdyby byl trénován na celém datasetu
- Slučování snižuje vliv zaujatosti a variance
- Sbor modelů má zpravidla stejnou zaujatost, ale nižší varianci než klasifikátor trénovaný na celém datasetu

Rozhodovací strom - bagging

- 500 stromů
- podobná zaujatost, ale menší variance
- obdobné množství chyb, ale rozhodovací hranice je pravidelnější



Bagging, nebo pasting

K rozhodnutí použijme křížovou validaci

Náhodné části (random patches), náhodné podprostory (random subspaces)

Náhodné části, náhodné podprostory

- Náhodné části
  - vzorkování trénovacích prvků i příznaků
  - -> ještě větší diverzita klasifikátorů
- Náhodné podprostory
  - vzorkování příznaků, použití všech prvků

Náhodné lesy (random forests)

- Sbor rozhodovacích stromů učený pomocí bagging

A. For each tree in the forest (for b = 1 to n_{trees}):
   a. Draw a bootstrap sample Z^* of size 1198 points
   b. Grow a decision tree based on the sample drawn from step a, by recursively repeating these steps for each node until the minimum node size $n_{min\text{-}node\text{-}size}$ or maximum depth $n_{max\text{-}depth}$ of tree is reached:
      a) Select $n_{dimentions}$ variables at random from the 5 predictor variables
      b) Using information gain calculation, pick the best split (variable-threshold pair, i.e. the certain value in certain variable that best splits)
      c) Split the node into two daughter nodes

B. Output the ensemble of trees $\{T_b\}^{ntrees}_1$. The majority votes will be taken for classification purpose.

Random Forest

Sklearn API:

```
RF = BaggingClasifier(
        DecisionTreeClassifier(splitter='rand
        om ),
        n_estimators=200, bootstrap=True
    )
```

Extrémně náhodné stromy (extra-trees, extremely randomized trees)

- Pro vytvoření práhu v každém uzlu se používá pouze podmnožina vstupních příznaků
- Rychlejší učení
  - Hledání vhodných prahů bývá nejdelší operací

Důležitost příznaků (feature importance, FI)

- Náhodné lesy měří relativní důležitost každého z příznaků
- Rychlé porozumění, jak jsou různé příznaky důležité
- FI: vážený průměr, kde je váha každého uzlu rovna množství trénovacích dat s uzlem asociovaných
- Sum(všechna FI) = 1

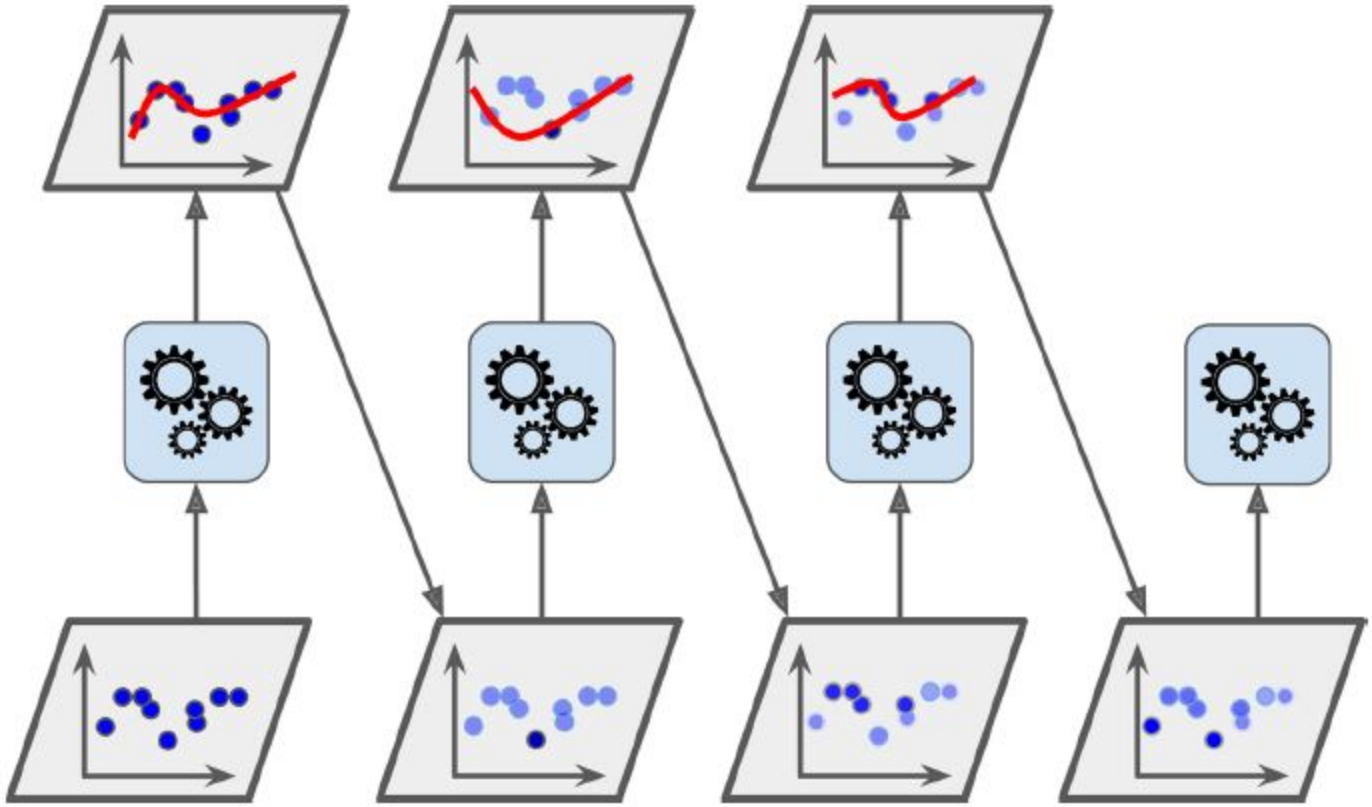
Boosting

Boosting

- Krokové vylepšení přesnosti modelu, kde při každém kroku vylepšujeme slabiny modelu v kroku předchozím
- Jedná se tedy o obecný algoritmus, nikoli o jeden model

AdaBoost

- Adaptivní Boosting
- Čím novější klasifikátor, tím více se zaměřuje na náročnější případy
- Zvyšují se relativní váhy nesprávně klasifikovaných příkladů
  - Váhy se *boostují*



AdaBoost

- Po dotrénování všech klasifikátorů sborové učení
- Klasifikátory mají rozdílné váhy v závislosti na jejich celkové přesnosti
  - Výsledný model: bagging či pasting

Vážená míra chyb

- Inicializace: $w^{(i)}$ rovno $1 / m$
- V dalším kroku:

$$r_j = \frac{\hat{y}_j^{(i)} \neq y^{(i)}}{\sum_{i=1}^{m} w^{(i)}}$$ where $\hat{y}_j^{(i)}$ is the j^{th} predictor's prediction for the i^{th} instance.

Váha klasifikátoru

$$\alpha_j = \eta \log \frac{1 - r_j}{r_j}$$

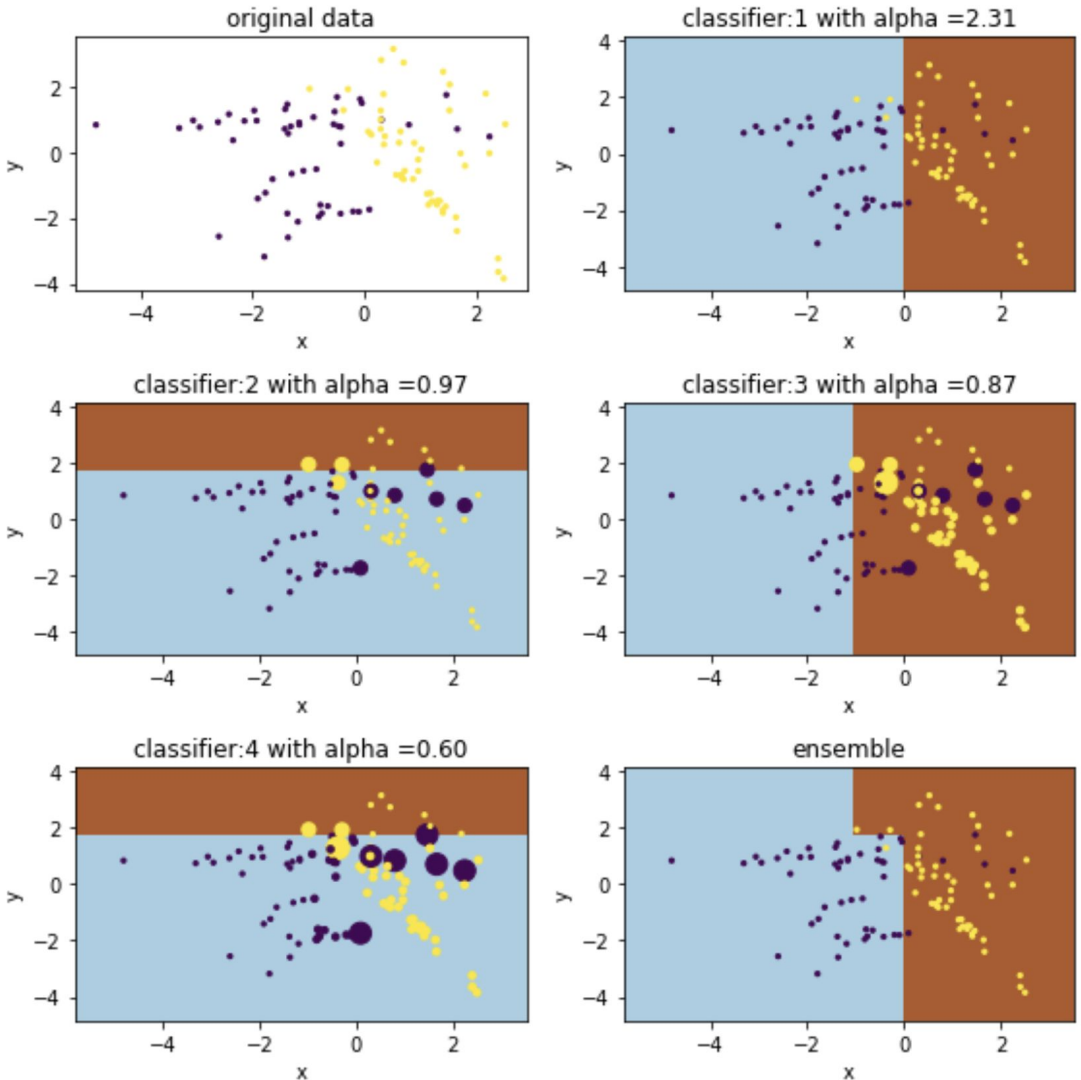
- kde η je faktor učení (learning rate)

Aktualizace vah

$$\text{for } i = 1, 2, \cdots, m$$

$$w^{(i)} \leftarrow \begin{cases} w^{(i)} & \text{if } \widehat{y_j}^{(i)} = y^{(i)} \\ w^{(i)} \exp\left(\alpha_j\right) & \text{if } \widehat{y_j}^{(i)} \neq y^{(i)} \end{cases}$$

- Všechny váhy jsou normalizovány
  - Poděleny sumou vah
- Algoritmus se zastaví, je-li dosaženo dostatečného počtu klasifikátorů nebo je-li nalezen dokonalý klasifikátor


original data
classifier:1 with alpha =2.31
classifier:2 with alpha =0.97
classifier:3 with alpha =0.87
classifier:4 with alpha =0.60
ensemble
x
y

Posílení gradientu (Gradient Boosting)

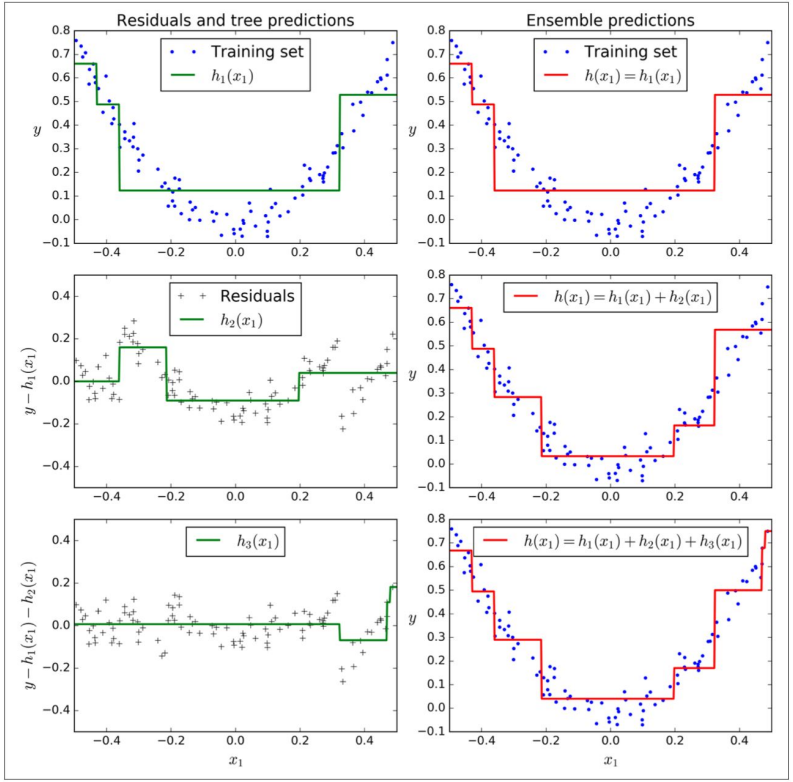
Posílení gradientu

- Sekvenční přidávání prediktorů;
- Každý opravuje svého předchůdce;
- Namísto úprav vah se snažíme novým prediktorem opravovat zbytkové chyby;

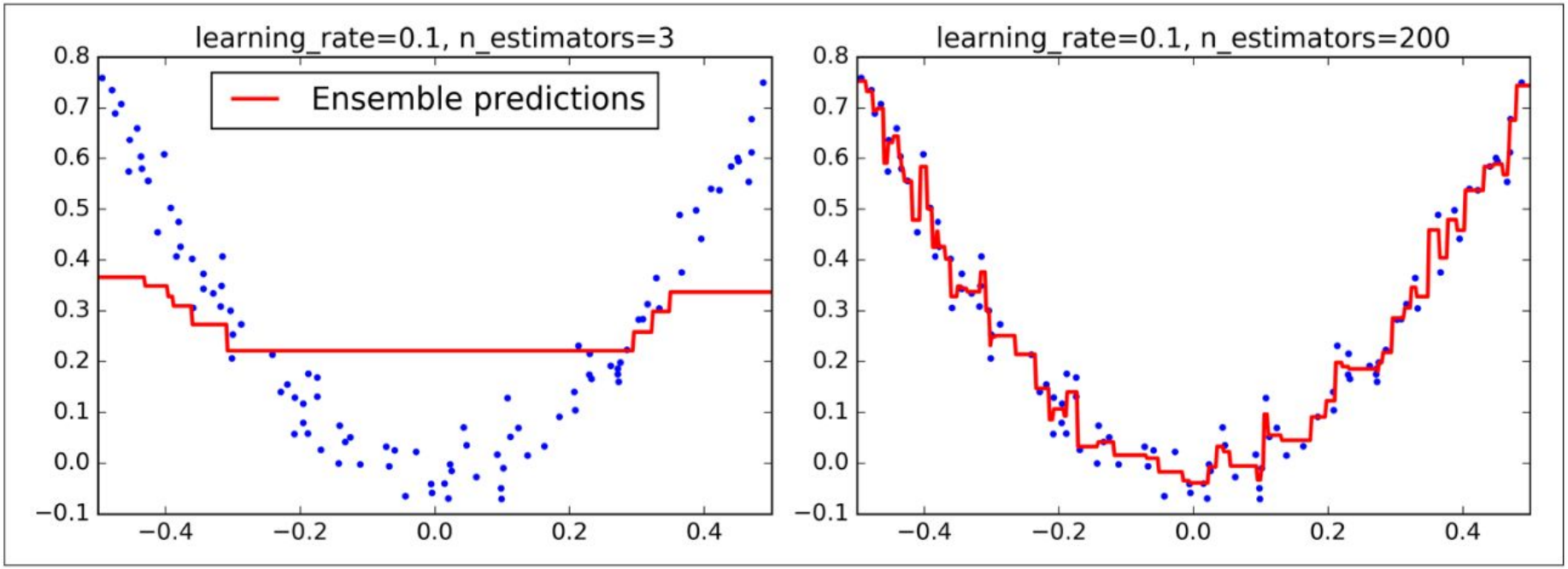
Posílení gradientu: Regresní stromy (Regression Trees)

```
# sklearn Gradient Boosting Regressor
from sklearn.tree import DecisionTreeRegressor

# iteration No. 1
tree_reg1 = DecisionTreeRegressor(max_depth=2)
tree_reg1.fit(X, y)

# iteration No. 2
y2 = y - tree_reg1.predict(X) # residuals
tree_reg2 = DecisionTreeRegressor(max_depth=2)
tree_reg2.fit(X, y2)

y3 = y2 - tree_reg2.predict(X) # residuals
tree_reg3 = DecisionTreeRegressor(max_depth=2)
tree_reg3.fit(X, y3)

# ensemble containing three trees
y_pred = sum(tree.predict(X_new) for tree in (tree_reg1, tree_reg2, tree_reg3))


# sklearn solution
from sklearn.ensemble import GradientBoostingRegressor
gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=3, learning_rate=1.0)
gbrt.fit(X, y)
```

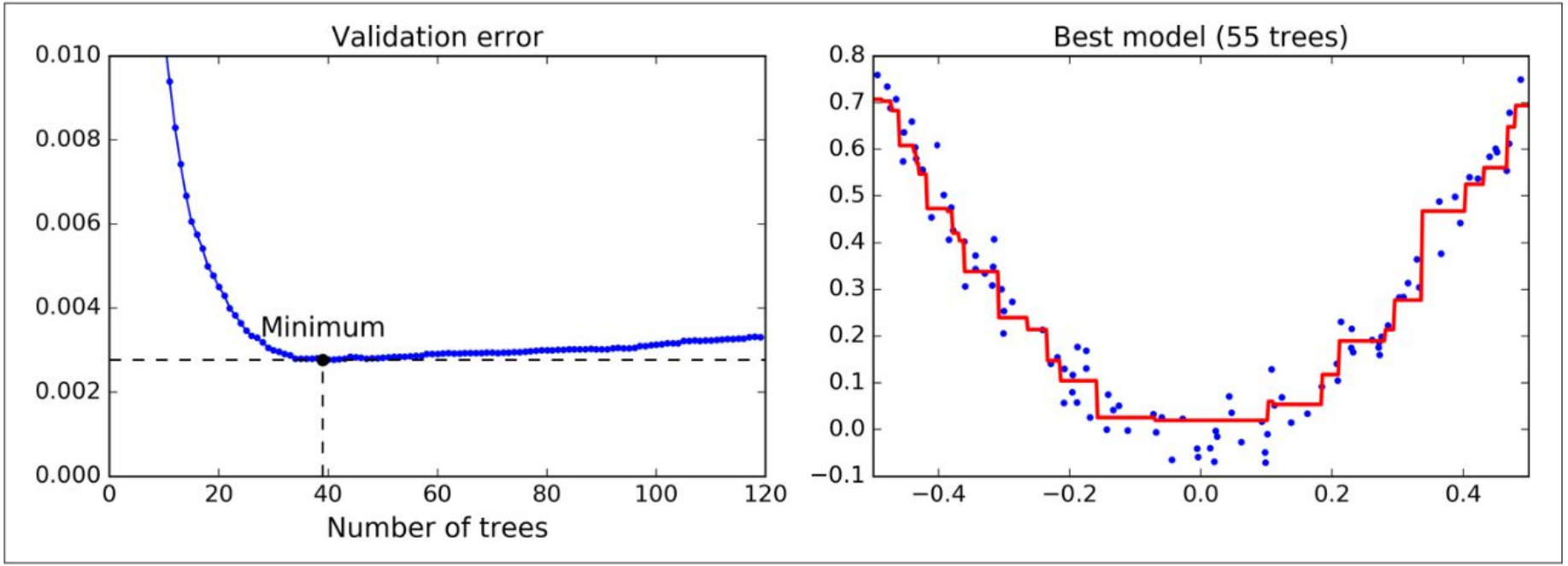
Posílení gradientu



Posílení gradientu



Posílení gradientu



Posílení gradientu RS

- Optimalizace: Klesající gradient (Gradient Descent);
- rychlost_učení (learning_rate) škáluje přínost každého stromu;
- Nízká hodnota, např. 0.1, potřebuje více stromů, ale obvykle se lépe generalizuje;

Posílení gradientu: Včasné / brzké / předčasné zastavení (Early Stopping)

```
# early stopping
gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=120)
gbrt.fit(X_train, y_train)

# staged_predict()
errors = [mean_squared_error(y_val, y_pred)
          for y_pred in gbrt.staged_predict(X_val)]
bst_n_estimators = np.argmin(errors)
```

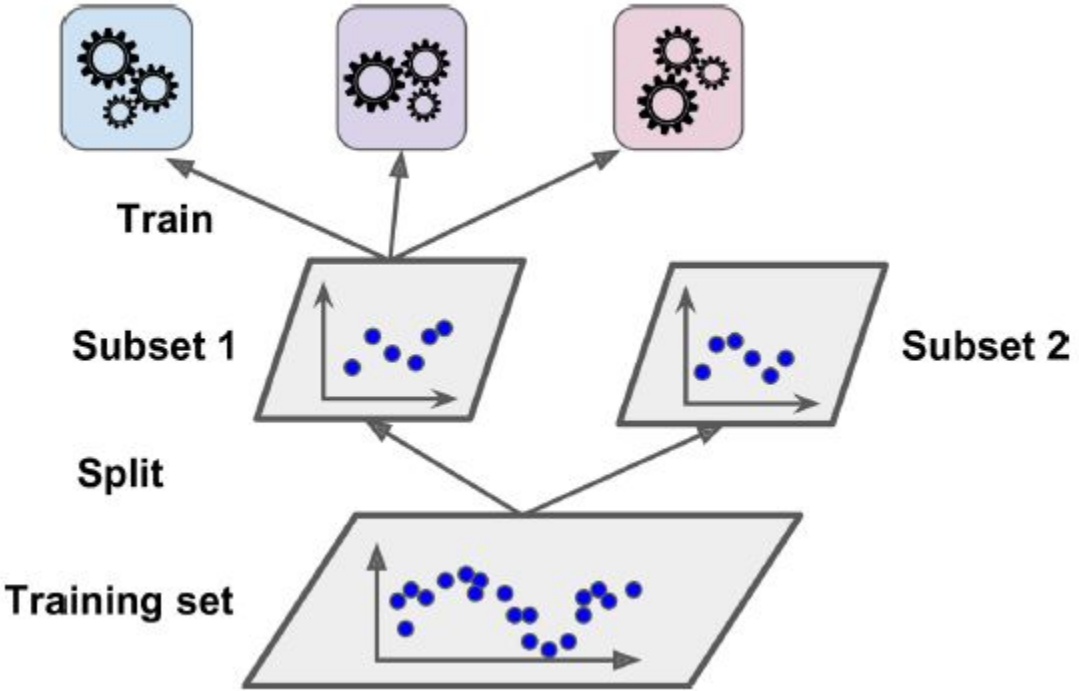
Předčasné zastavení

```
# early stopping
gbrt = GradientBoostingRegressor(max_depth=2, n_estimators=120)
gbrt.fit(X_train, y_train)

# staged_predict()
errors = [mean_squared_error(y_val, y_pred)
          for y_pred in gbrt.staged_predict(X_val)]
bst_n_estimators = np.argmin(errors)
```

```
# Actual early stopping (5)
# use warm_start=True
gbrt = GradientBoostingRegressor(max_depth=2, warm_start=True)
min_val_error = float("inf")
error_going_up = 0
for n_estimators in range(1, 120):
    gbrt.n_estimators = n_estimators
    gbrt.fit(X_train, y_train)
    y_pred = gbrt.predict(X_val)
    val_error = mean_squared_error(y_val, y_pred)
    if val_error < min_val_error:
        min_val_error = val_error
        error_going_up = 0
    else:
        error_going_up += 1
        if error_going_up == 5:
            break # early stopping!
```

Stochastické posílení gradientu

Trénování první vrstvy



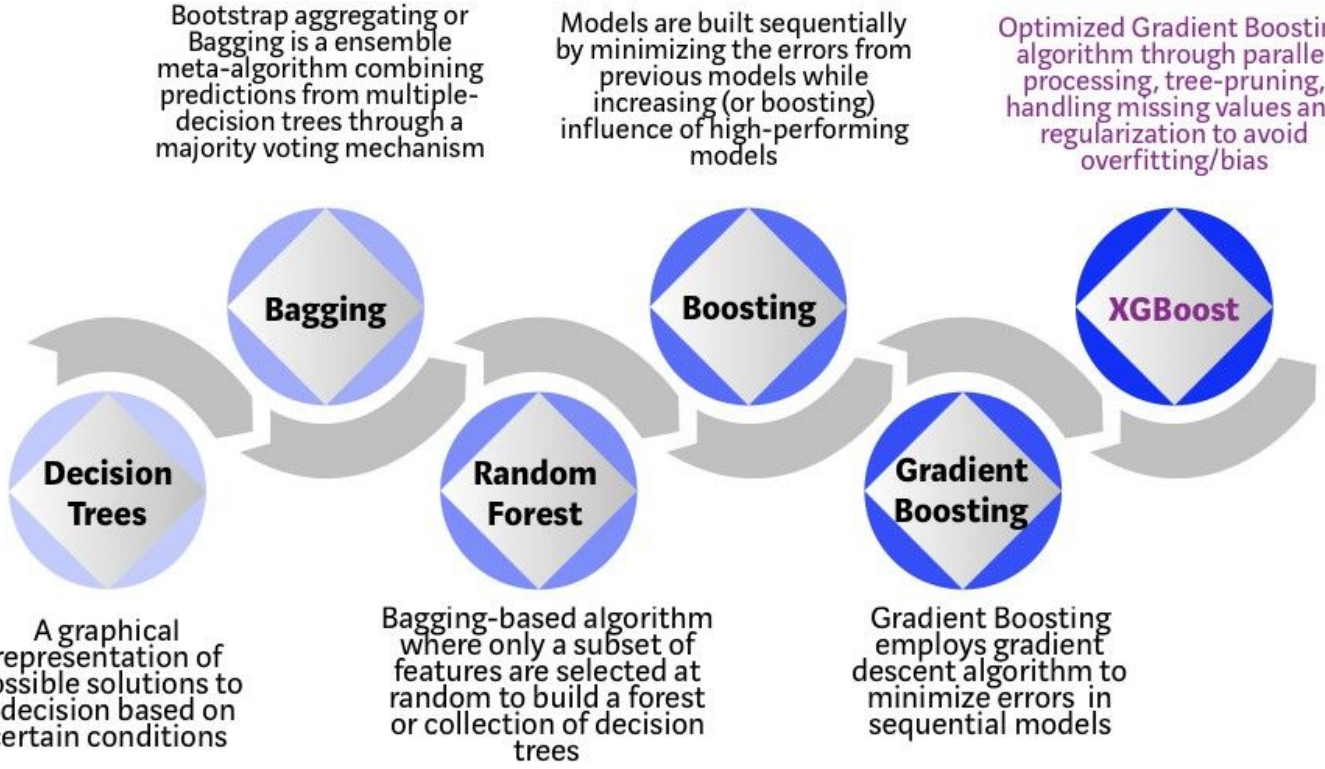
Stochastické posílení gradientu

- **hyperparametr subsample (podvzorek)**: udává podíl trénovacích vzorků použitý pro každý strom;
- **subsample**=0.25

  -> model bude trénován na 25 % náhodně vybraných vzorků;

  + zrychluje trénování;

  - vyšší bias, nižší rozptylu;

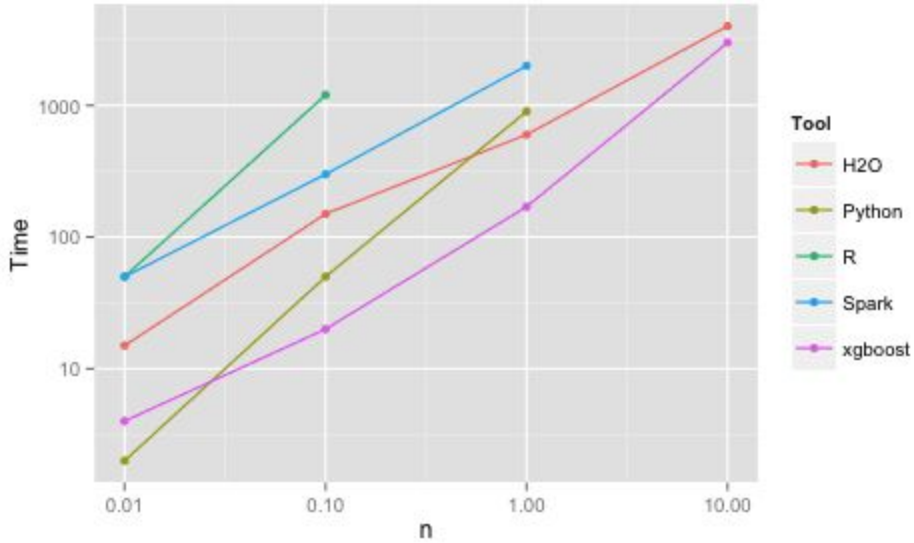

Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

XGBoost

- Cíl: posunout hranice výpočetních zdrojů pro posílené rozhodovací stromy;
- V poslední době dominuje v oblasti aplikovaného strojového učení a soutěží Kaggle;

Proč XGBoost

1. Rychlost;
2. Výkon;

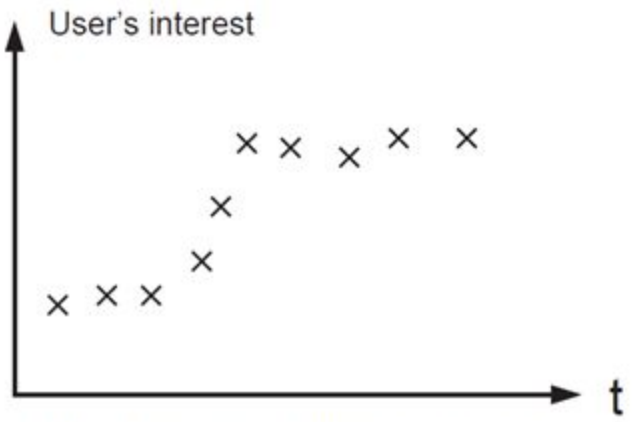


XGBoost

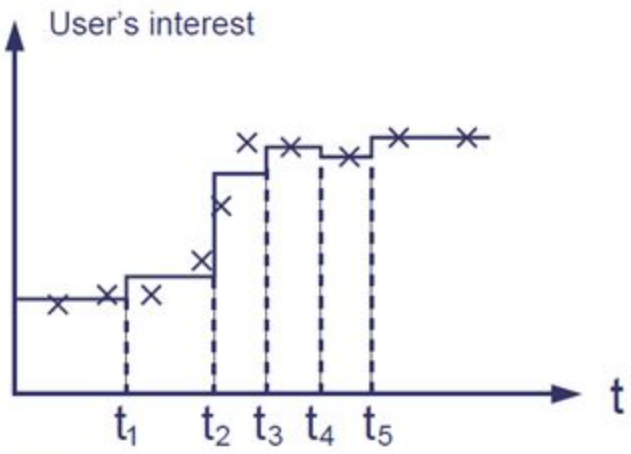
- Funkce:
  obj(theta) = L(theta) + O (theta)

  ztrátová_funkce + *regularizace*

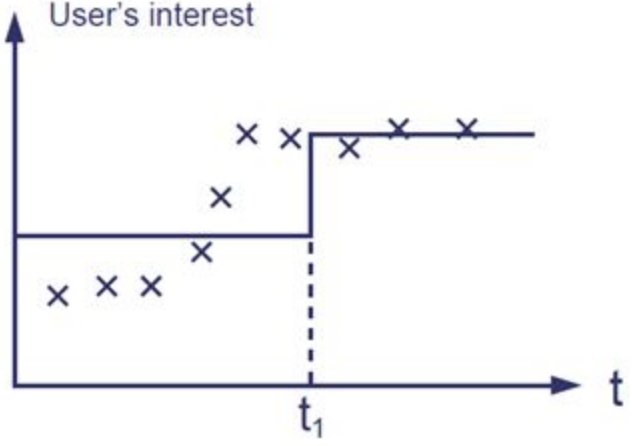
- L: jak dobře model predikuje/detekuje (MSE)
- O: Složitost modelu



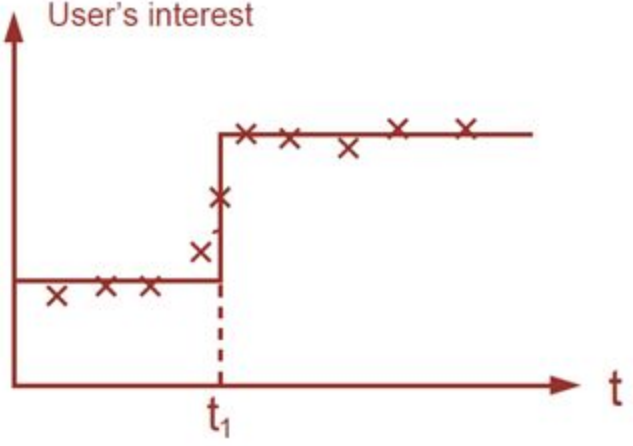
Observed user's interest on topic k against time t



☒ Too many splits, $\Omega(f)$ is high



☒ Wrong split point, $L(f)$ is high



☑ Good balance of $\Omega(f)$ and $L(f)$

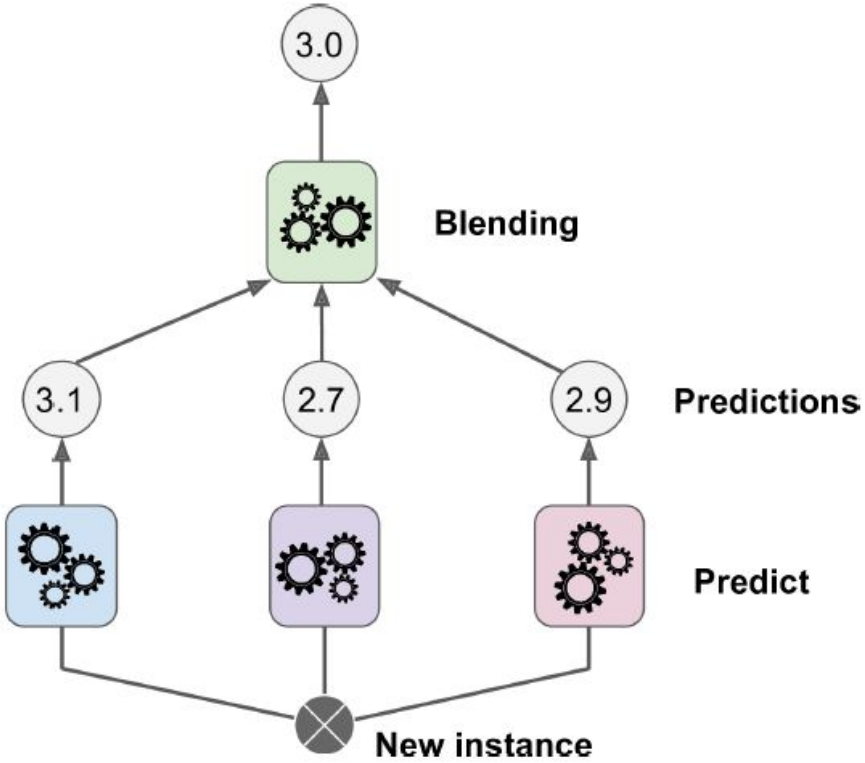
- Algoritmy pro posílení představují odlišný pohled na strojové učení: přeměňují slabý model na silnější, aby napravily jeho slabiny!

Stohování (Stacking)

Stohování

- Stacking je zkratka pro stacked generalization
- Agregace všech “prediktorů prediktorů” v ansámblu
- Obsahuje tzv. mixér (blender) nebo metažák (meta-learner)
  - Vstupem mu jsou jednotlivé predikce, na jejichž základě vytvoří predikci závěrečnou
  - Pro jeho trénování se používá hold-out set

Agregace výsledků v mixéru

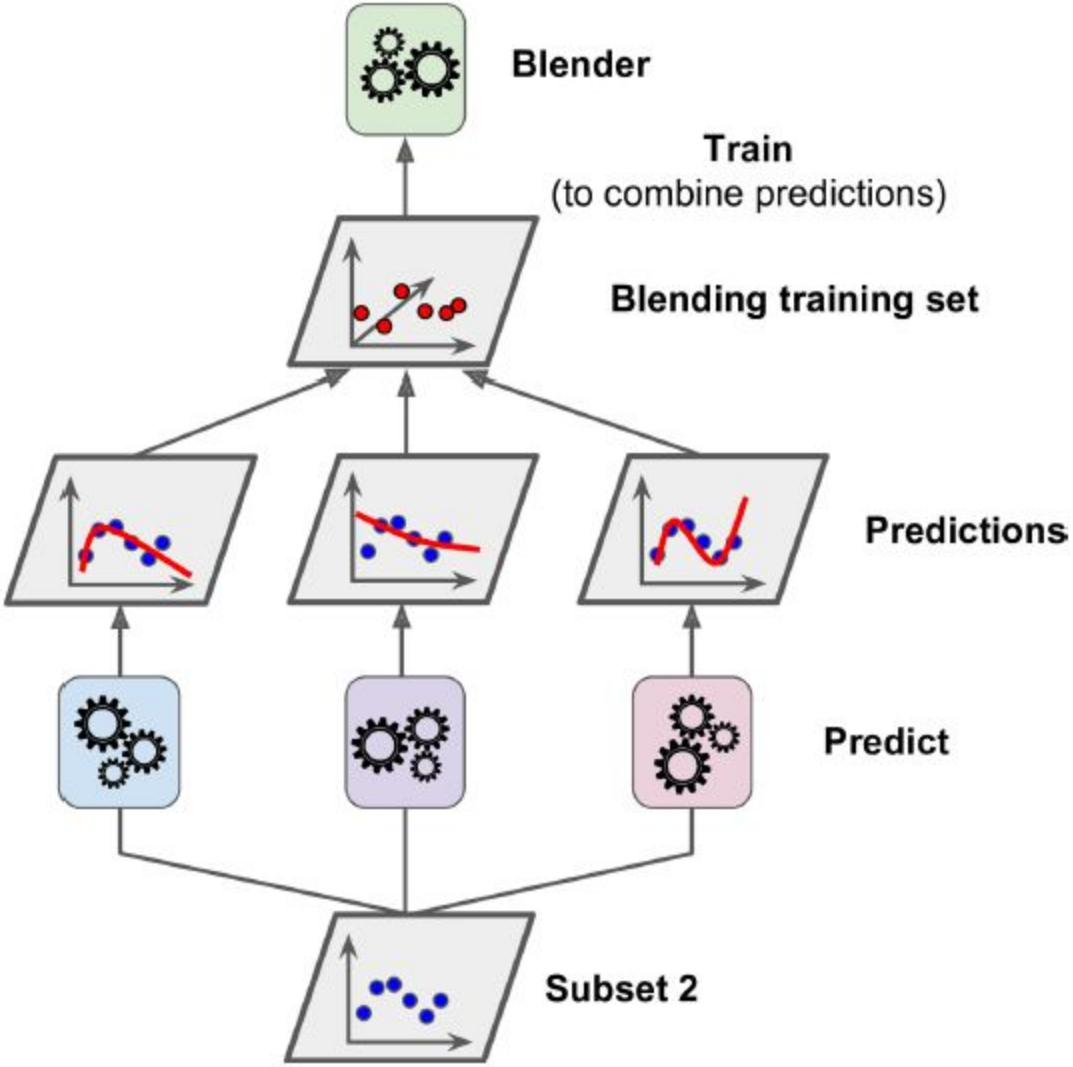


Hold-out set

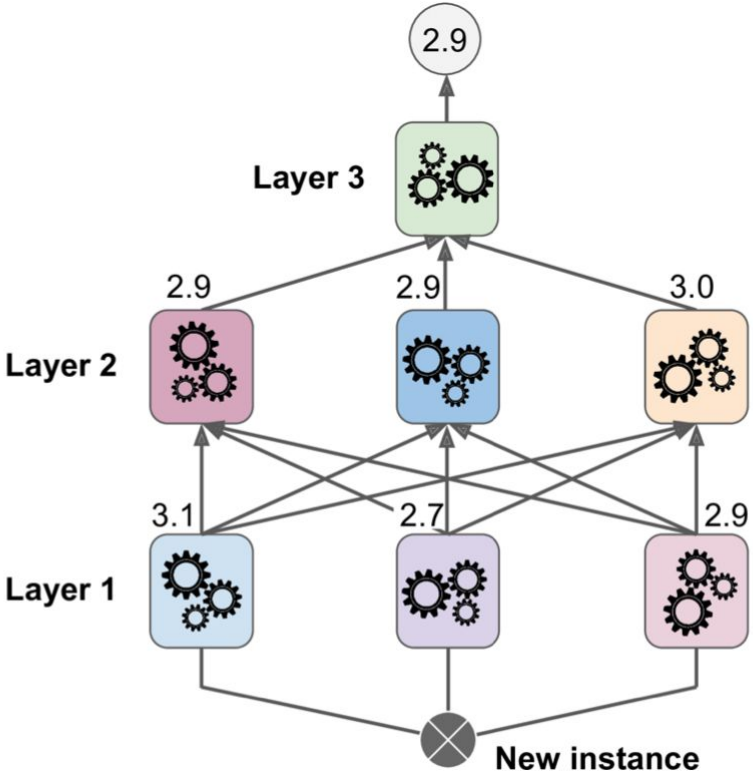
- Používá se 2 podsad trénovacích dat;

1. Podsada 1: trénování prediktorů první vrstvy
2. Prediktory první vrstvy se použijí k vytvoření predikcí na druhé sadě (hold-out set)
   a. Predikce jsou ‘čisté’
3. Vytvoří se nová trénovací sada použitím predikovaných hodnot jako vstupů a původních výstupů
4. Mixér se trénuje na nových trénovacích datech

Tedy: Mixér se učí predikovat cílovou třídu z výsledků první vrstvy


Blender
Train
(to combine predictions)
Blending training set
Predictions
Predict
Subset 2

Vícevrstvé stohovací ansámbly



- Rozhodovací strom
- Hlasovací klasifikátor
- Bagging (s opakováním)
- Pasting (bez opakování)
- Náhodné části (náhodné prvky a příznaky)
- Náhodné podprostory (náhodné příznaky)
- Náhodný les (bagging, nejlepší příznak v náhodné podmnožině)
- Extrémně náhodné stromy (náhodný práh)
- Boosting (zlepšování slabých míst)
- AdaBoost (zvyšuje relativní váhy nesprávně klasifikovaných případů)
- Včasné zastavení
- Posílení gradientu (naprav zbytkové chyby)
- Stochastické posílení gradientu (malé náhodné podmnožiny)
- XGBoost (regularizované stochastické posílení gradientu)
- Stohování (mixér = metažák)

- Decision Tree
- Voting Classifier
- Bagging (without replacement)
- Pasting (with replacement)
- Random patches (random instances, features)
- Random Subspaces (uses random features)
- Random forest (bagging, best feature among random subset)
- Extra trees (random thresholds)
- Boosting (improving weaknesses)
- AdaBoost (weighting where wrong predictions made)
- Early stopping
- Gradient Boosting (fit residual errors)
- Stochastic Gradient Boosting (small random subsets)
- XGBoost (regularized Stochastic Gradient Descent)
- Stacking (blender = meta learner)

Otázky?