Ensemble Methods, Instance based learning

Applied machine learning (EDAN95)
Lecture 04 — Decision trees, ensemble methods, and clustering
2019–11–13
Elin A. Topp

Goodfellow chapter 3, Géron chapter on DTs
Mitchell chapter 3
various sources

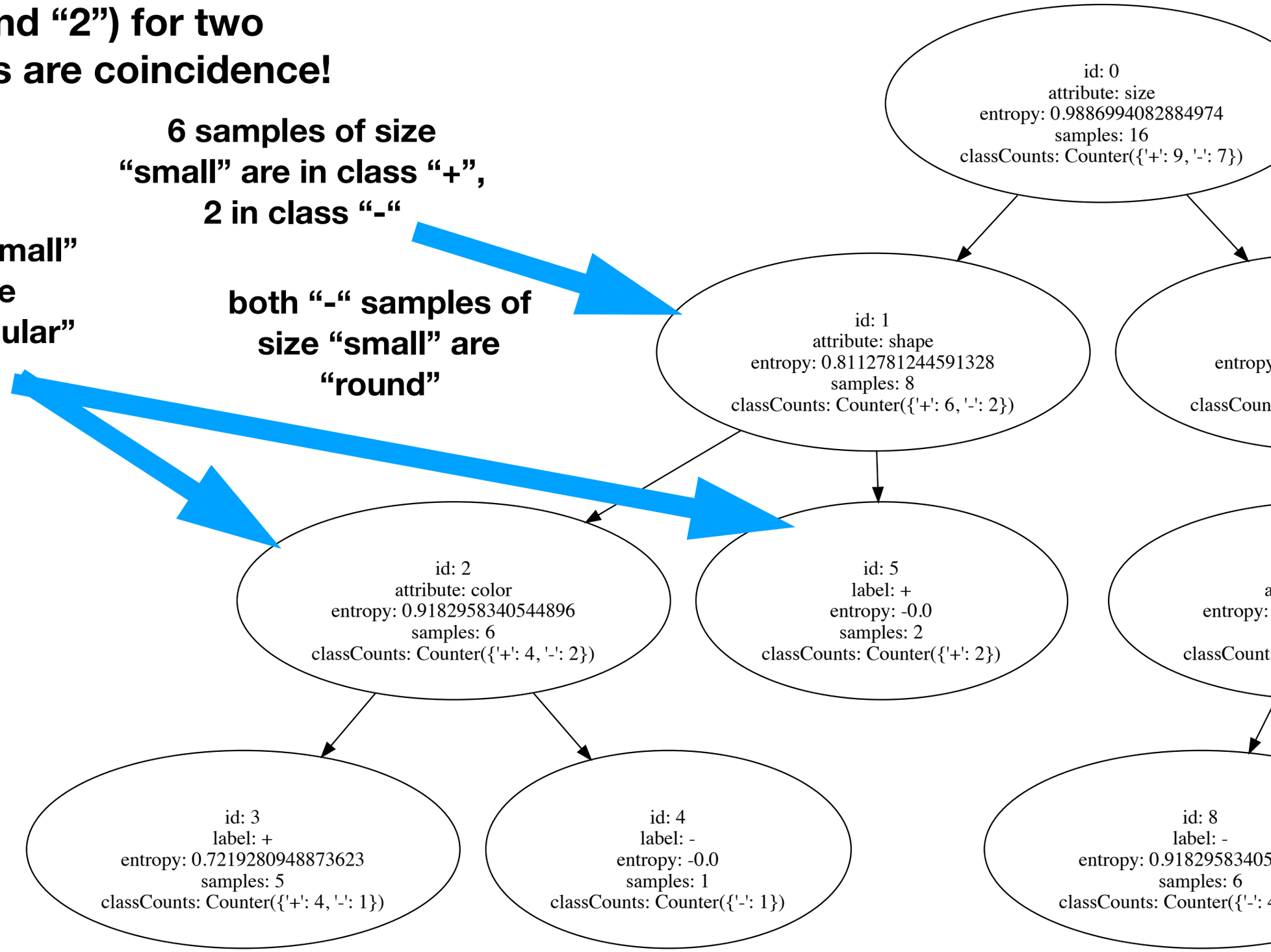
ID3-Decision Tree based on maximum Information Gain

Numbers (“6” and “2”) for two different distinctions are coincidence!

6 samples of size “small” are in class “+”, 2 in class “-“

6 samples of size “small” are also of shape “round”, 2 are “irregular”

both “-“ samples of size “small” are “round”



Today's agenda

- Decision trees - what could (possibly) go wrong?
- Ensemble methods / Random Forests
- k-NN classification
- Outlook Clustering (k-Means)

Today's agenda

- Decision trees - what could (possibly) go wrong?
- Ensemble methods / Random Forests
- k-NN classification
- Outlook Clustering (k-Means)

Issues with Decision Trees

- Consider a new example with which you want to modify your tree…
- Consider a very unbalanced data set (like the concept learning example)
- Consider really unseen attribute values in examples …
- …

Issues with Decision Trees

- Non-existing values for certain attributes during training
  (OBS, not to confuse with attribute values that are not represented in the sample set)
  - Missing values are replaced by the most common value in the considered set for the given attribute
- Continuous values
  - Use decisions not based on equality, but on intervalls (see the SciKitLearn implementation)
- Natural bias of Information Gain towards attributes with many values, tree grows too specific
  - Limiting breadth of the tree by looking at groups of values (intervals)
  - Using other methods to find the best split attribute than Information Gain
- Overfitting the data (actually an issue with most methods in ML)
  - Pruning (going through the tree and taking out subtrees that do not contribute to performance, i.e., the new tree performs no worse than the original one on the validation data)
  - Limiting the depth / number of splits by setting constraints on the minumum number of samples for a split or in a leaf

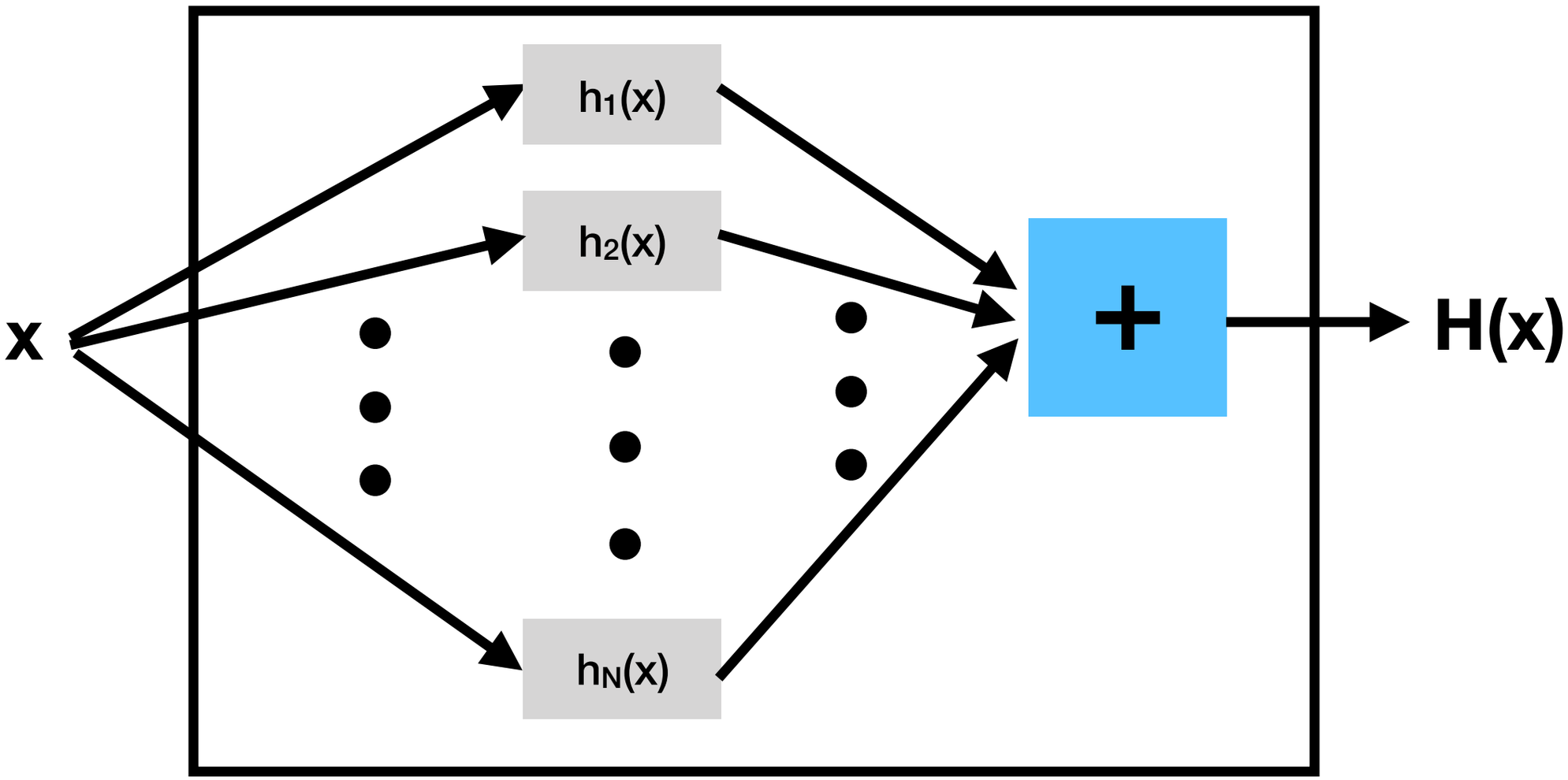
Today's agenda

- Decision trees - what could (possibly) go wrong?
- Ensemble methods / Random Forests
- k-NN classification
- Outlook Clustering (k-Means)

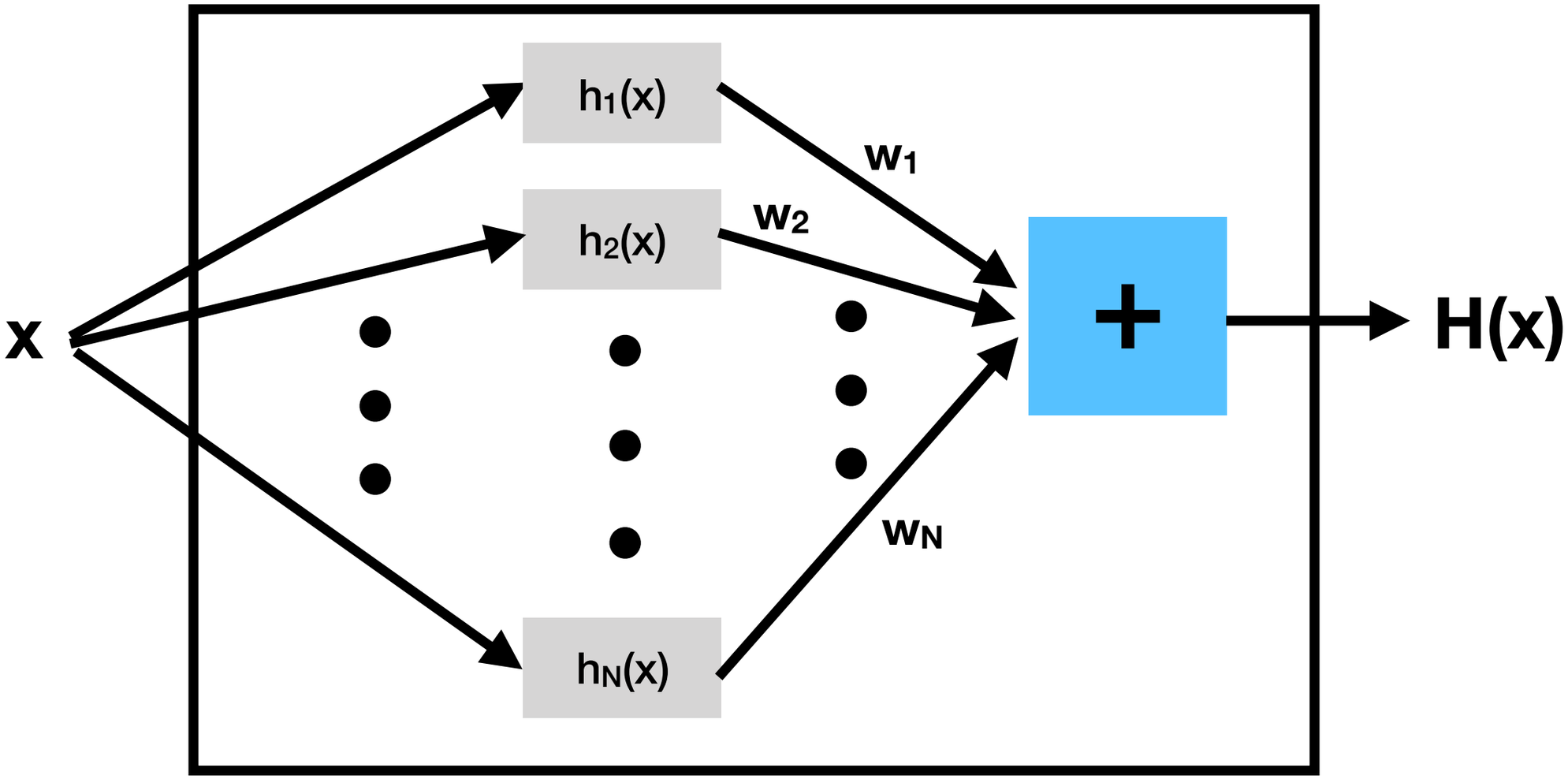
Single opinion vs collective decision

- One single classifier (we assume a decision tree for now, but it can be anything) can make a mistake, but hopefully that happens with a likelihood below random
- Several single classifiers—if trained with some variation—will most likely make different mistakes
- If there is some sense in the classifiers (mistake less likely than random!), there will be a majority of correct / sensible answers in the crowd.
- Train *N* classifiers (not one tree, but a forest) and have them somehow come to a collective conclusion
- Several ways of producing the final output, given a set of hypotheses $\{h_i\}$ for the answer from each of the *N* classifiers, $i = 1,\ldots,N$

Method 1: Averaging / Voting

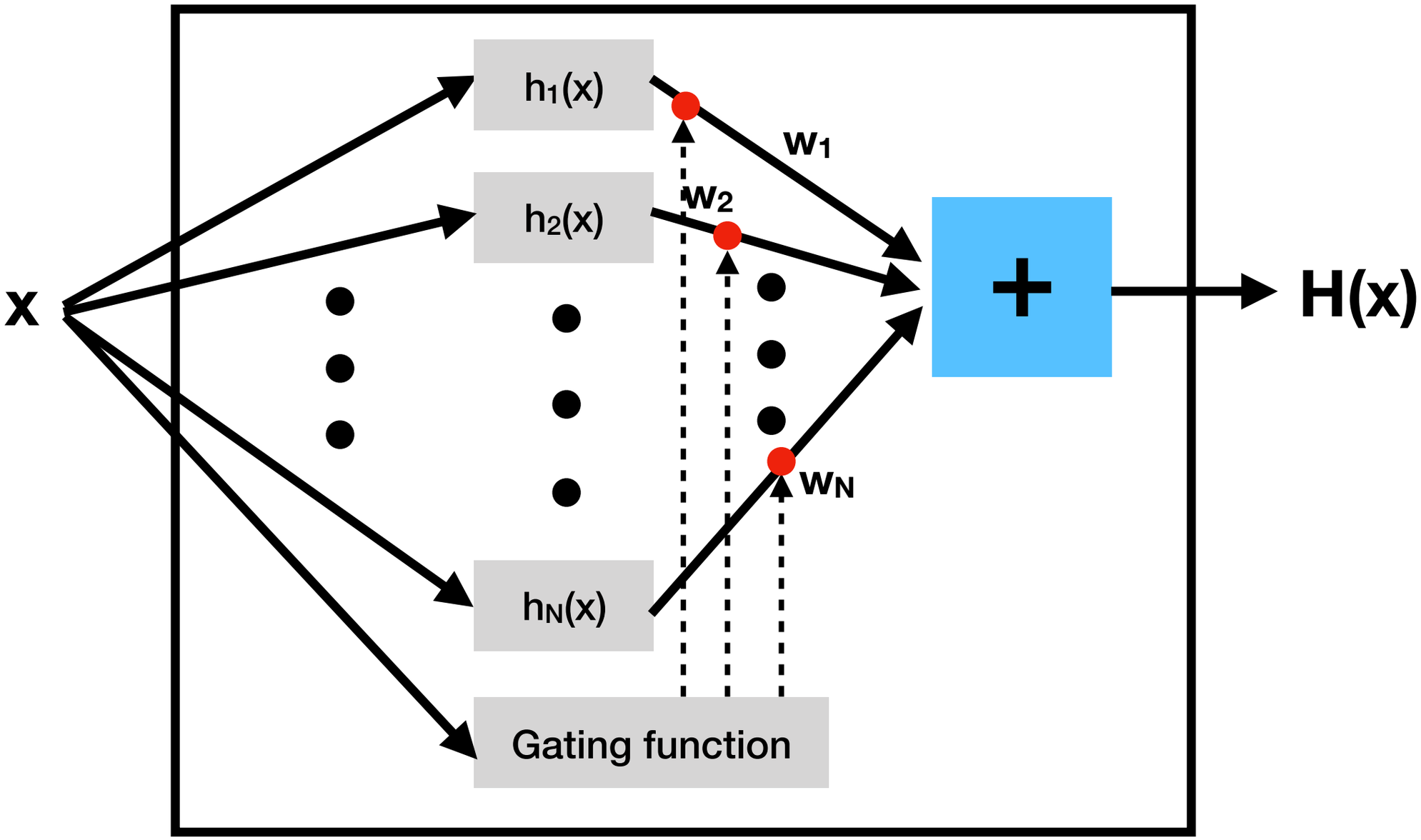


Method 2: Weighted Averaging / Voting



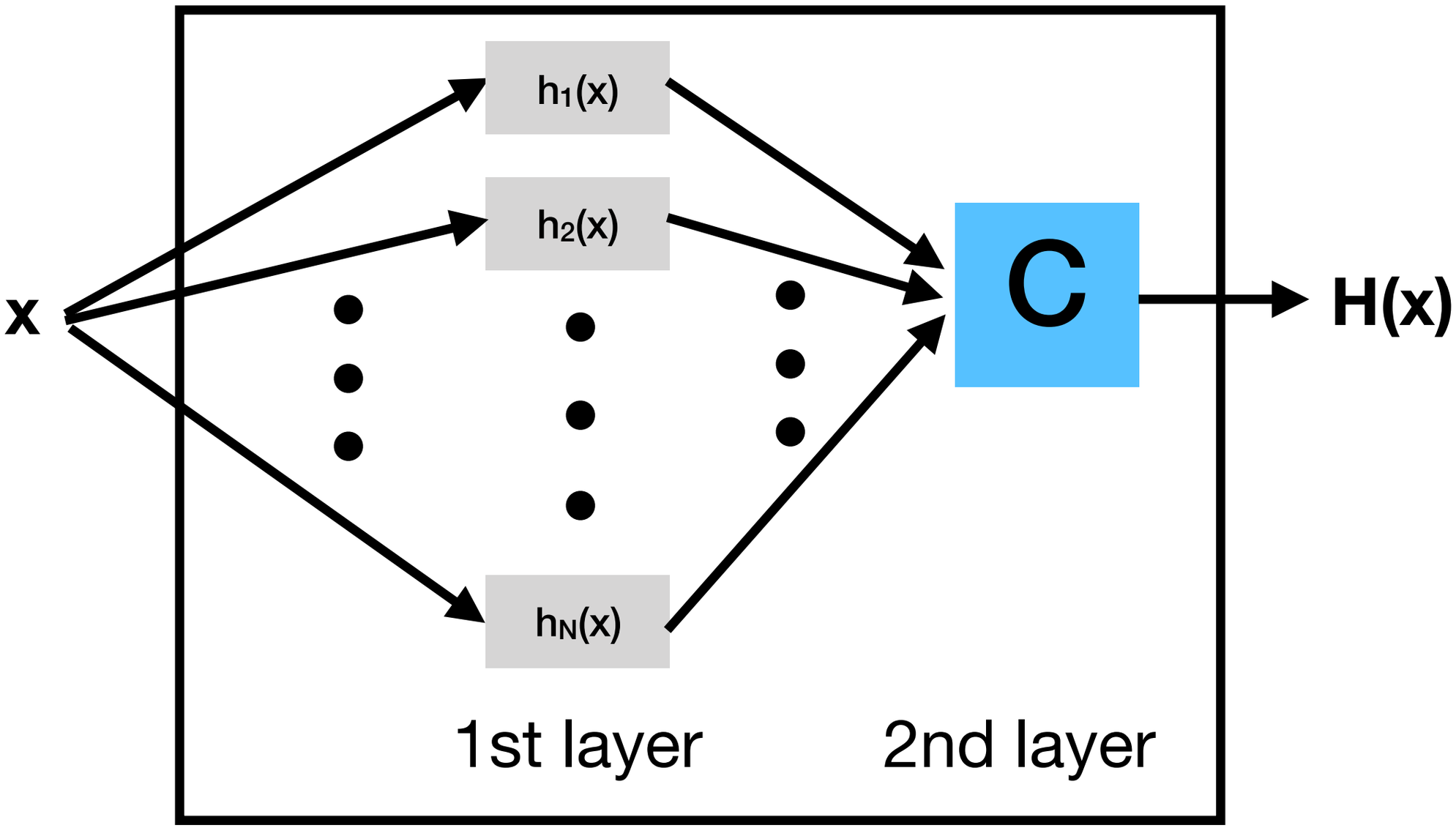
Slide based on material from Eric Eaton (see lecture slides EDAN95 2018, lecture 4)

Method 3: Gating



Slide based on material from Eric Eaton (see lecture slides EDAN95 2018, lecture 4)

Method 4: Stacking

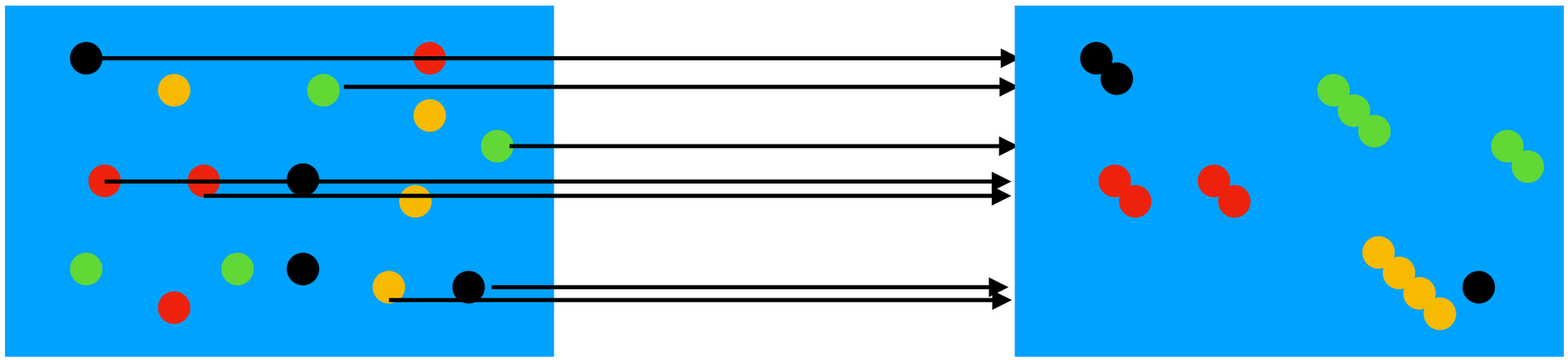


Slide based on material from Eric Eaton (see lecture slides EDAN95 2018, lecture 4)

Creating several different classifiers

- There might be different causes for the mistakes your original single classifier makes.
  - Difficult samples (sometimes, reality is nasty): No recipe, you will have to live with this, and be aware that there are these uncertainties
  - Overfitting (even after modelling quite carefully): Vary the training sets
  - Noise in the data / some features: Vary sets of input features

Manipulating training data: Bootstrap replication



Exclude some (30%) of data from the bootstrapping

Bagging (Bootstrap AGGregatING)

- Do a Bootstrap replicating round, create thus N training sets
- Train N classifiers, one on each set
- Estimate performance on the out-of-bootstrap data (the ~30%)
- Combine output according to previously suggested methods

Boosting (e.g., AdaBoost)

- Improves (weak) classifier(s) over time (steps $t = 1,\ldots,T$) by emphasizing mispredicted instances (in the training set!)
- Assume classes $y_i \in \{-1,1\}$ and classification hypotheses $h_t(x_i) \in \{-1,1\}$
- Initially, all instances are of the same importance, e.g.

$$w_{t,i} = \frac{1.0}{|samples|} \quad for \quad t = 0$$

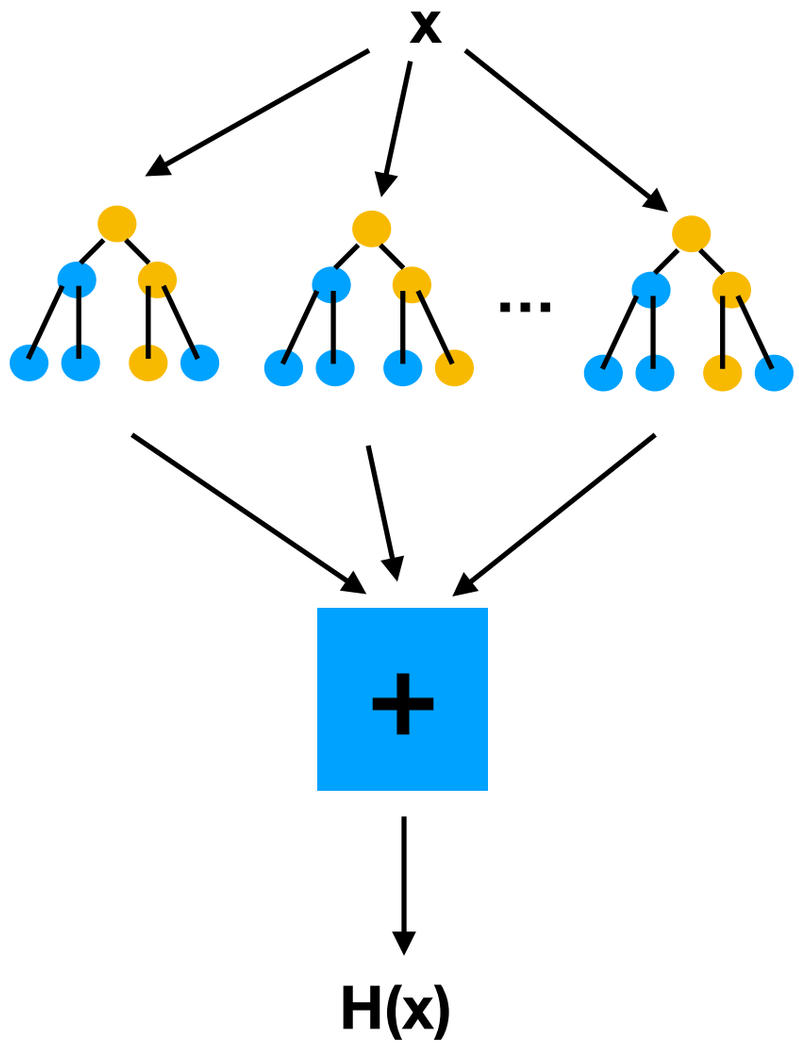
- In each iteration, compute the weighted training error (opposite of accuracy) ϵ (each instance x_i in the training set contributes with its weight to the error, misclassified instances, where $h_t(x_i) \neq y_i$ obviously more than correctly classified ones)
- Update the weights so that a wrongly classified sample gets a higher weight

$$w_{t+1,i} = w_{t,i} \cdot e^{-\beta_t y_i h_t(x_i)} \quad with \quad \beta_t = \frac{1}{2} ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

- normalise the weights, and iterate
- when done, produce hypothesis as $H(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \beta_t h_t(\mathbf{x})\right)$
- or, when done with the boosting, retrain your classifier, considering now the weighted instances

Random Forest

- In principle, a bagging approach:
  - Do a Bootstrap replicating round, create thus *N* training sets
  - Train *N* DT classifiers, one on each set, BUT
    - Use only a randomly picked set of attributes for each DT
    - Do not prune the trees and estimate performance on the out-of-bootstrap data (the ~30%)
  - Combine output according to previously suggested methods (averaging)



Today's agenda

- Decision trees - what could (possibly) go wrong?
- Random Forests / ensemble methods
- k-NN classification (instance based learning)
- Outlook Clustering (k-Means)

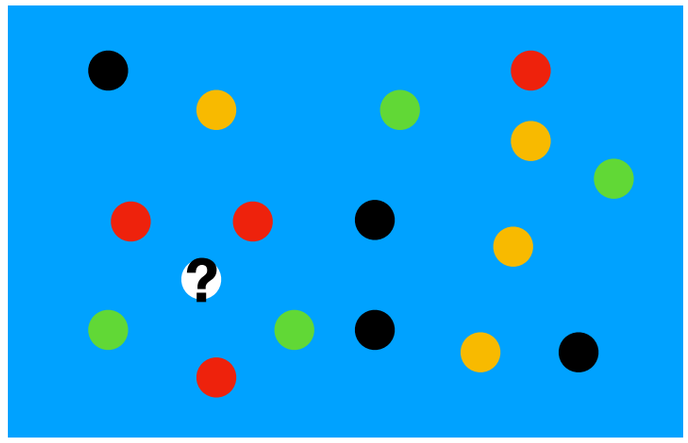
Instance based learning

- Naive case (brute force algorithm):
  - Do not actually fit a model, other than representing the training data as points in the feature (attribute) space
  - For each test sample, find the closest point(s) in the training set and classify the test sample accordingly (goes through ALL data points in each case)

- Less brute force:
  - create some model, e.g., fit a "decision tree" that allows to restrict the search space

k-Nearest Neighbour

- k-Nearest Neighbour classifier averages (votes) over the k closest points



? = ●

Today's agenda

- Decision trees - what could (possibly) go wrong?
- Random Forests / ensemble methods
- k-NN classification (instance based learning)
- Outlook Clustering (k-Means) → more in Lectures 10 / 11

Outlook on lectures 5-9

- Feedforward networks, loss, back propagation, optimizers, evaluation, …
- CNNs
- RNNs
- LSTMs and GRUs
- Autoencoders, GANs

Outlook on lab / assignment 3 (CNNs)

- Lab session as usual (work and present in groups of two)
- Report, to be handed in ~ a work week after you have been passed for the implementation (exact deadlines will be specified with the instructions)
  - needs to be INDIVIDUALLY written
  - should follow the instructions carefully
  - is to be handed in at https://sam.cs.lth.se/portal
    (OBS, this is somewhat of a beta-version of a system to hand in material, bear with us)
  - must be submitted by a single author (even if the system would allow to add partners!)
  - might be **ignored if not submitted according to the instructions**
- It is possible to hand in one (1) delayed report in January, exact deadline will follow - depends on whether there will be an extra lab session, but earliest date for late hand-in will be:
  January 17, 2020

Today's summary

- Discussed potential problems with decision trees
- Showed some ensemble methods
- Introduced k-NN Classifier

- Reading:
  - Géron, Hands-on ML, Material on Github (spec Decision Trees)
  - Lecture slides lecture 4, 2018
  - Mitchell, chapter 3, Decision Trees