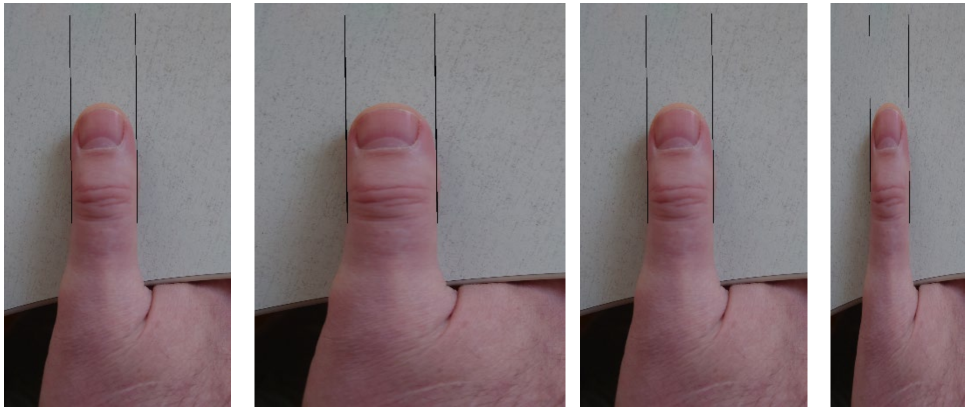
Ensemble learning

Machine Learning (ML)

Agenda

- Motivation
- Bias and Variance revisited
- Bagging
- Random Forests
- Boosting
- Stacking

Motivation



= 1 inch (2,54 cm)

Motivation

“If each member of a jury is more likely to be right than wrong, then the majority of the jury, too, is more likely to be right than wrong; and the probability that the right outcome is supported by a majority of the jury is a (swiftly) increasing function of the size of the jury, converging to 1 as the size of the jury tends to infinity.” - Marquis de Condorcet, 1785

Bias and Variance revisited

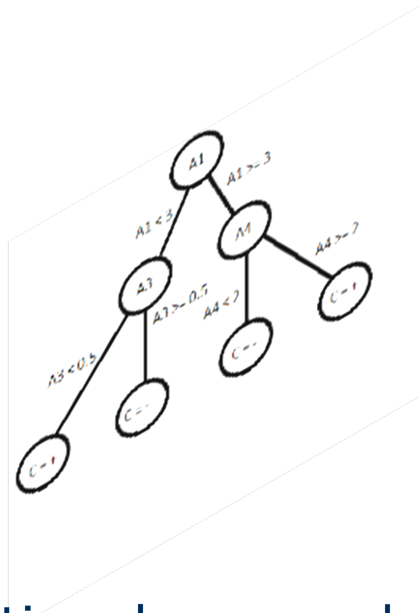
- Given a **target** function f
- **Find** a hypothesis h, such that
  - Such that $h \approx f$
- **Bias** is the **tendency** of h to **deviate** from expected value when averaged on different training sets
  - High bias, linear model
  - Low bias, k-nearest neighbor
- **Variance** is the **amount** of change in h **due** to **fluctuations** in training data
  - Low variance, linear model (stable)
  - High variance, decision tree (unstable)
- **Bias-variance tradeoff**, the **choice** between a more **complex** low bias h vs **simple** low variance h

Ensemble learning

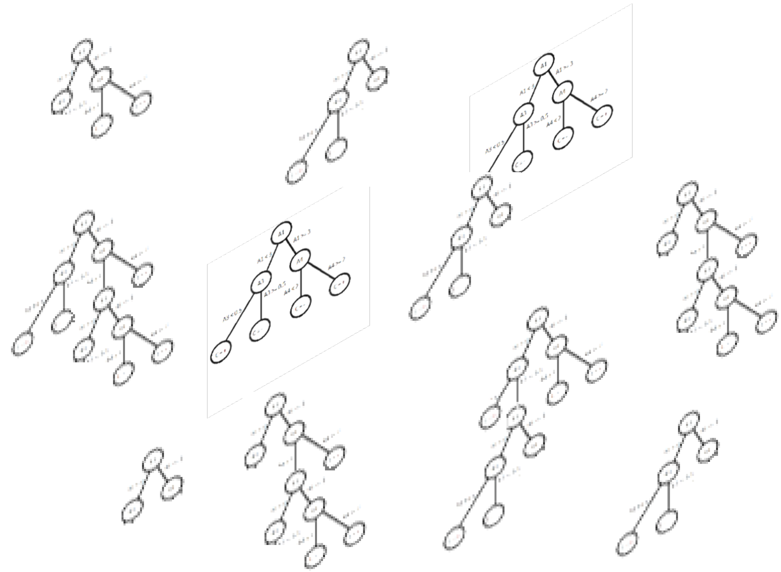
- **Successful** ensembles require **diversity**
  - Classifiers should make **different** mistakes

1. **Adjusting** induction method
   - By using **different** base **learners** (with different biases)
     - This will most **probably** ensure that they make **different errors**
2. Adjust **data** by using **homogenous** induction method
   - Bagging
   - Boosting
3. **Adjust** data **and** selection method
   - Random Forest

Ensemble learning



versus



Traditional approach

– Build one **good** model

Ensemble approach

– Build **many** models and average the results

Bagging

- Idea: Combining **many unstable** predictors to produce a **stable** predictor, by reducing **variance**

- A bootstrap sample *B* (one bag) of a set of examples *E* is created by **randomly selecting** $n = |E|$ examples from *E* **with replacement**

- The **probability** of an example in E **appearing** in B is

$$1 - \left(1 - \frac{1}{n}\right)^n = 1 - \frac{1}{e} \approx 0.632$$

- Examples in $E \setminus B$ is called **out–of-bag** examples
  - i.e. examples not in B

Bootstrap example

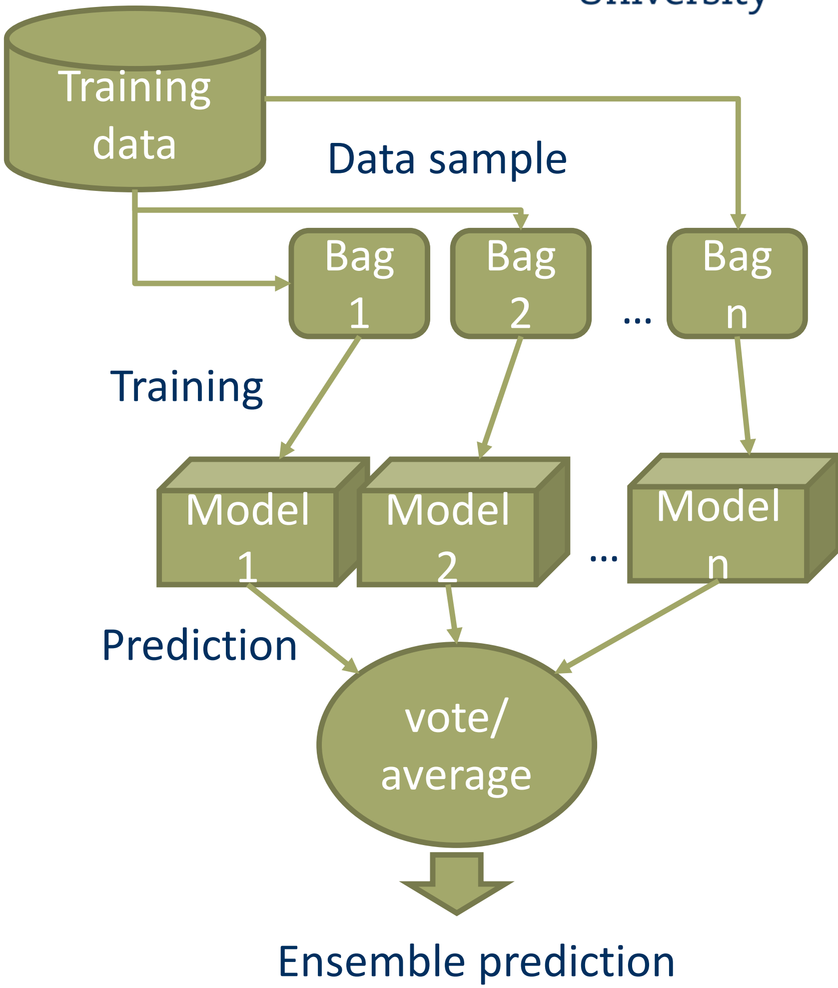
```
>>>from sklearn.utils import resample
>>>e = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>>samp = resample(e)
>>>samp
array([1, 4, 8, 1, 5, 2, 8, 2, 3, 9])
>>>samp = resample(e)
>>>samp
array([3, 7, 7, 9, 2, 4, 4, 5, 2, 9])
>>>samp = resample(e)
>>>samp
array([1, 7, 2, 4, 8, 8, 8, 2, 1, 8])
>>>samp = resample(e)
>>>samp
array([9, 3, 9, 1, 6, 6, 9, 8, 6, 9])
>>>samp = resample(e)
>>>samp
array([6, 9, 6, 8, 5, 0, 0, 5, 9, 5])
```

Bagging

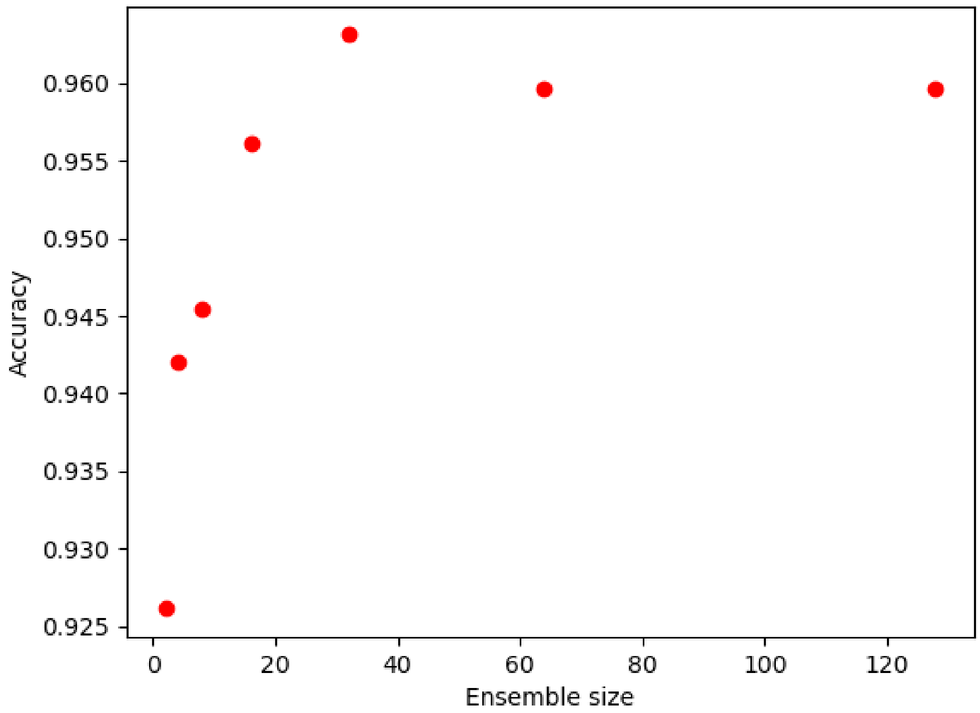
- Take repeated **bootstrap samples** from training set D
- Bootstrap sampling
  - Given training set *D* containing *N* training examples
    - Create *D'* by drawing *N* examples at random with replacement from *D*

- Bagging*
  - Create k bootstrap samples $B_1, \ldots, B_k$
  - Train a classifier (or regressor) on each B_i
  - Predict new instance
    - by majority vote (classification)
    - by averaging value (regression)

* L. Breiman. 1996. Bagging predictors. Machine Learning, 24(2):123-140



Bagging example result



Breast-cancer-Wisconsin dataset from UCI repository
Stratified 10 fold cross validation
Decision trees as base classifier

Bagging tips

- **Any** base learner can be used **but**,
  - good results are obtained by unstable (**high variance**) learners like
    - Decision trees
    - Neural networks

- Works in similar way for both classification and regression, only the **voting** scheme needs to change

- Using decision trees, **pruning** should **not** be used,
  - as this **decrease** the variance

- Can be used **without** separate **validation** or **test set**
  - **Out-of-bag** examples can be used for this

Random Forest

- Idea: Combine **bagging** and **the random subspace method (RSM)^**(feature bagging), in one method to further increase diversity (variance) in the base learner to produce a low **variance** predictor (ensemble).

- Similarly to bagging RSM samples features with replacement, typically the sqrt(no_features) is used

- No_features = 10, $\sqrt{10}$ = 3 features selected at random

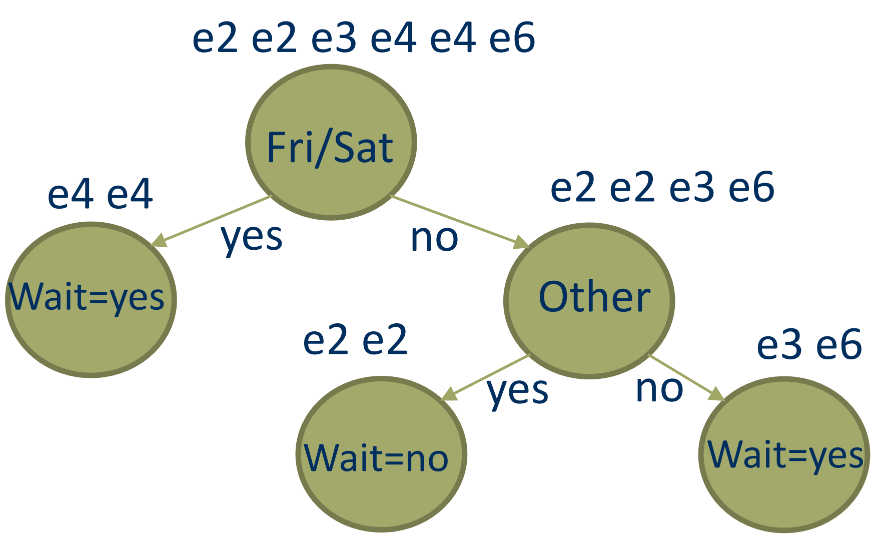
^T. K. Ho. 1998. The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8):832-844

Feature selection example

```
>>> from sklearn.utils import resample
>>> import numpy as np
>>> e = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> no_features = int(np.sqrt(len(e)))
>>> samp = resample(e, n_samples=no_features)
>>> samp
array([7, 6, 3])
>>> samp = resample(e, n_samples=no_features)
>>> samp
array([9, 3, 1])
>>> samp = resample(e, n_samples=no_features)
>>> samp
array([8, 0, 5])
>>> samp = resample(e, n_samples=no_features)
>>> samp
array([9, 2, 4])
>>> samp = resample(e, n_samples=no_features)
>>> samp
array([8, 3, 1])
```

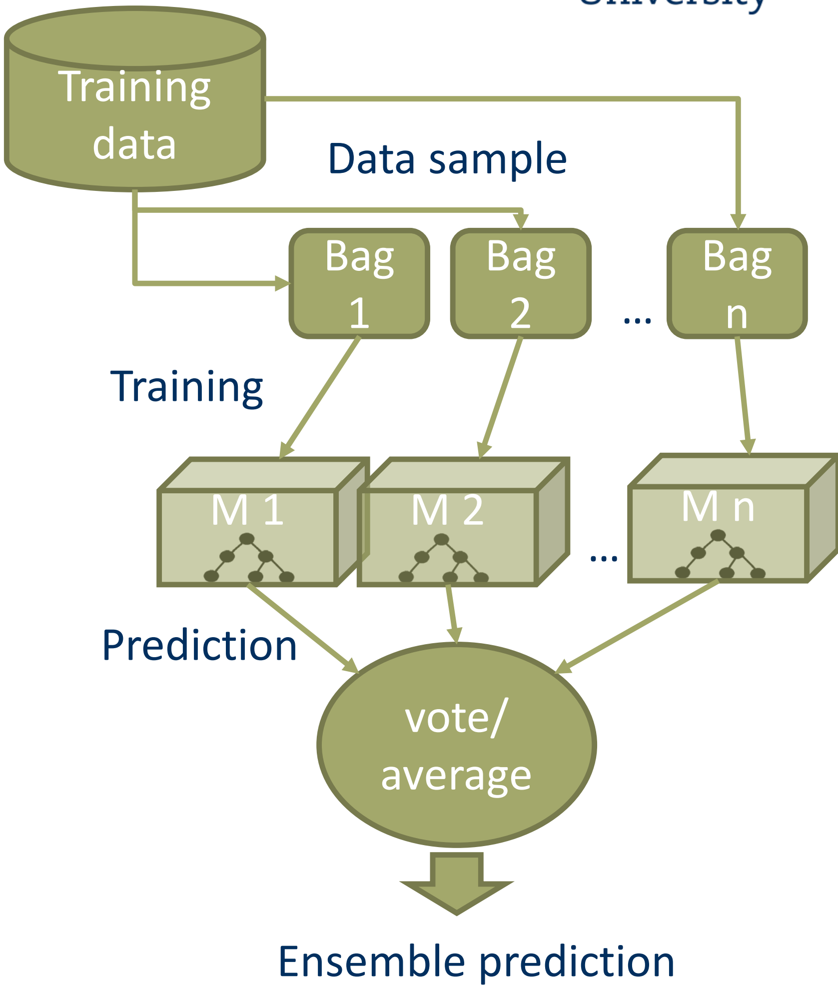
Feature selection example

Ex.	Other	Bar	Fri/Sat	Hungry	Guests	Wait
e2	yes	no	no	yes	full	no
e2	yes	no	no	yes	full	no
e3	no	yes	no	no	some	yes
e4	yes	no	yes	yes	full	yes
e4	yes	no	yes	yes	full	yes
e6	no	yes	no	yes	full	yes

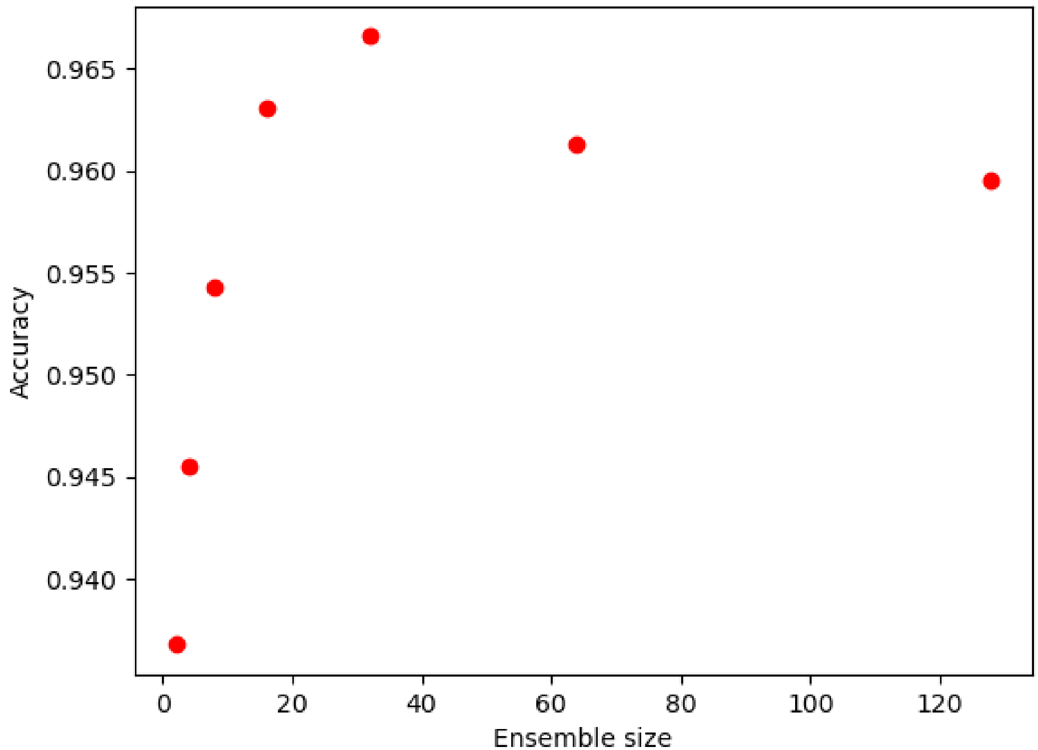


Random Forest

- Construct decision trees on bootstrap replicas
  - **Restrict** the node **decisions** to a small **subset** of **features** picked **randomly** for each node
- Do not prune the trees
  - Estimate tree performance on **out-of-bootstrap** data
  - Approximately **35%** of data in a bootstrap sample
  - Predict new instance
    - by majority vote (classification)
    - by averaging value (regression)



Random forest example result



Breast-cancer-Wisconsin dataset from UCI repository
Stratified 10 fold cross validation
Decision trees as base classifier

Stockholm University

Random forest tips

- In (Fernandez-Delgado et al 2014), an **empirical** investigation was presented using:
  - 179 classifiers from 17 families, incl. all standard approaches
  - 121 datasets, incl. all of UCI except the largest ones
  - The random forest versions **ranked** the highest and they obtained near to the best accuracy for almost all the data sets

- It was **concluded** that
  - The classifiers most likely to be the best are the random forest (RF) versions

M. Fernandez-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? Journal of Machine Learning Research, 15(1), pp. 3133-3181, 2014.

Randomization

- Repeated **application** of a base learner that contains some **stochastic** element (or that can be adjusted to **contain** stochastic elements) such as
  - the setting of **initial** weights in a neural network
  - **sampling** of grow and prune data
  - choice of **attribute** to split on in decision trees

- Randomization may be **combined** with other techniques

Boosting - AdaBoost

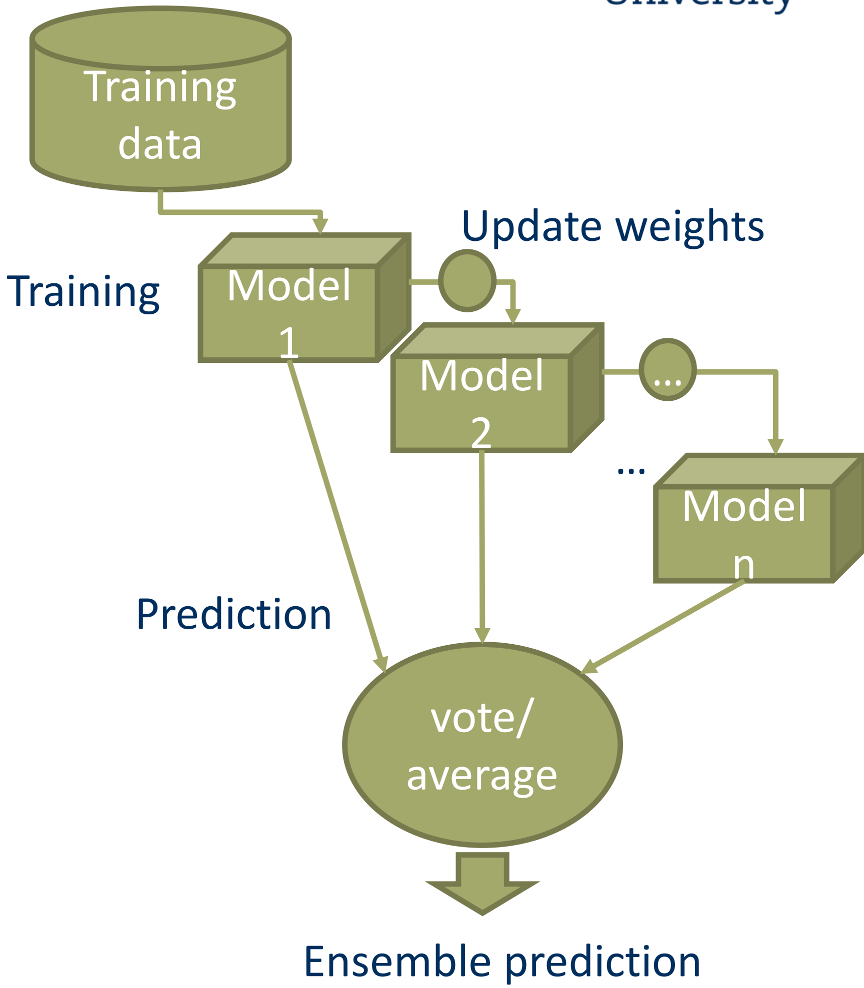
- **Ada***ptive* **Boost***ing* -> AdaBoost
- Idea: Turn a **weak stable** learner into a **strong** learner by focusing on the difficult cases to predict.

- Each predictor is created by using a **biased** sample of the **training** data, hence boosting minimize the bias of the ensemble (and also variance).

- General Steps in boosting
  1. Train a **weak** model on some training data
  2. Compute the **error** of the model on each training example
  3. Give **higher** importance to **examples** on which the model made **mistakes**
  4. Re-train the model using **re-weighted** training example

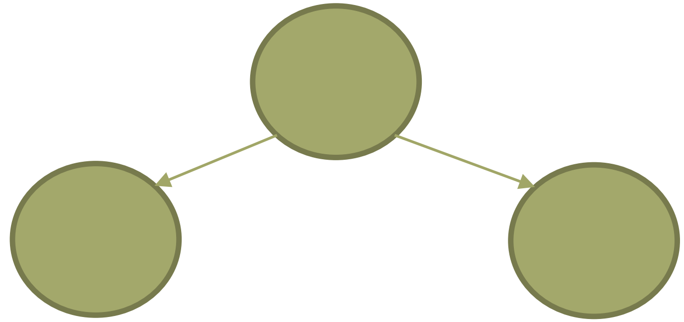
AdaBoosting

- An **sequential** procedure to **adaptively change** the **distribution** of training data by
  - Focusing more on previously **miss-classified** examples
    - Initially all examples are assigned **equal** weights
    - After a classifier C_i is learned
      - The weights are **adjusted** to **increase** weights of **misclassified** examples of C_i
      - Using **accuracy** on a validation set as the **basis** for **re-weighting**



Y. Freund and R. Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. J. Comput. Syst. Sci. 55(1): 119-139

AdaBoost

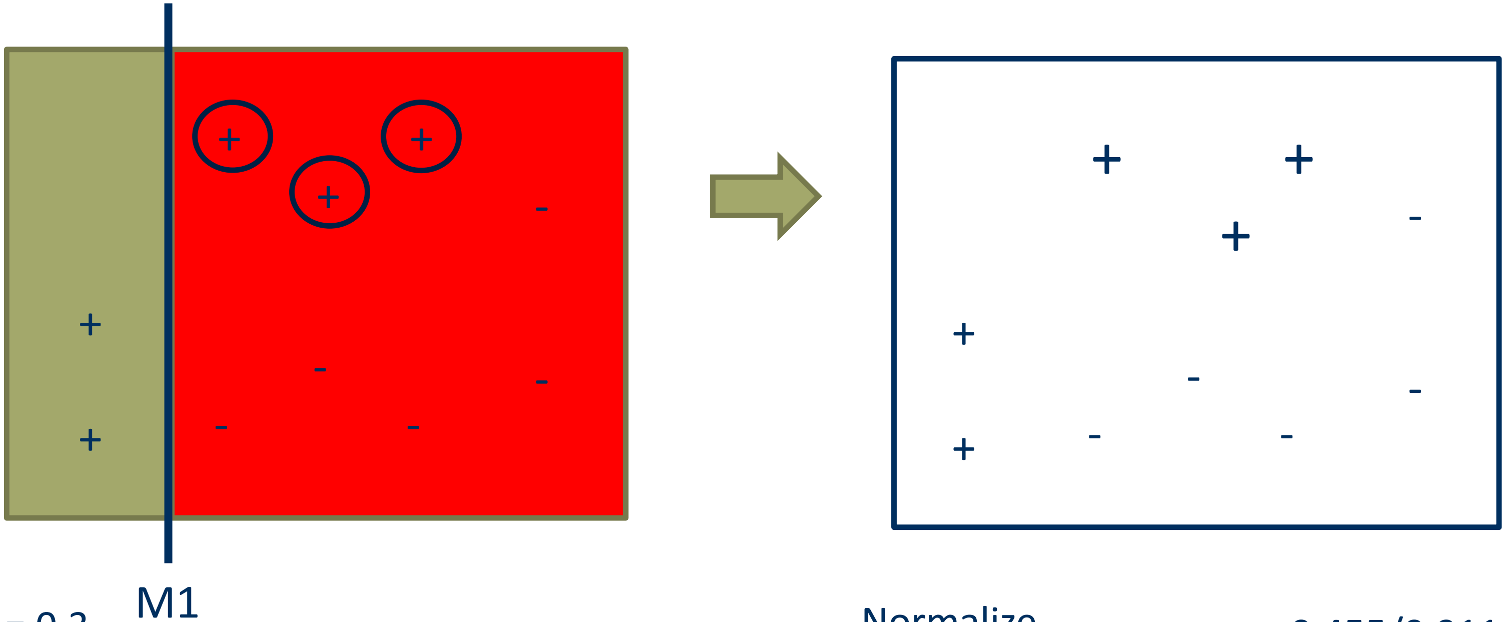


- **Weak stable** learner – decision stump

```
Input: instances e1,..., em, base learner BL, size S
Output: a set of model-weight pairs M
w1, ..., wm = 1
M = {}
for i = 1 to S:
      Mi = BL({(e1,w1), ...,(em,wm)})
      Err = (w1*err(Mi,e1)+...+wm*err(Mi,em))/(w1+...+wm)
      if Err = 0 or Err > 0.5 then break
      αi = ½ ln((1-Err)/Err)
      for j = 1 to m:
            if err(Mi,ej) = 0 then wj = wj*e(-αi) #down-w
            else wj = wj*e(αi)                     #up-weighted
      Normalize(w1, …, wm)
      M = M + (Mi, αi)
```

Stockholm University

AdaBoost - Example



Err = 0.3

αi = ½ ln((1-Err)/Err) = ½ ln((1-0.3)/0.3) = **0.42**

wj = wj*e(-0.42) = 0.065 #down-weighted

wj = wj*e(0.42) = 0.152 #up-weighted

Normalize

0.065 * 7 = 0.455

0.152 * 3 = 0.456

0.455 + 0.456 = 0.911

0.455/0.911 = 0.5

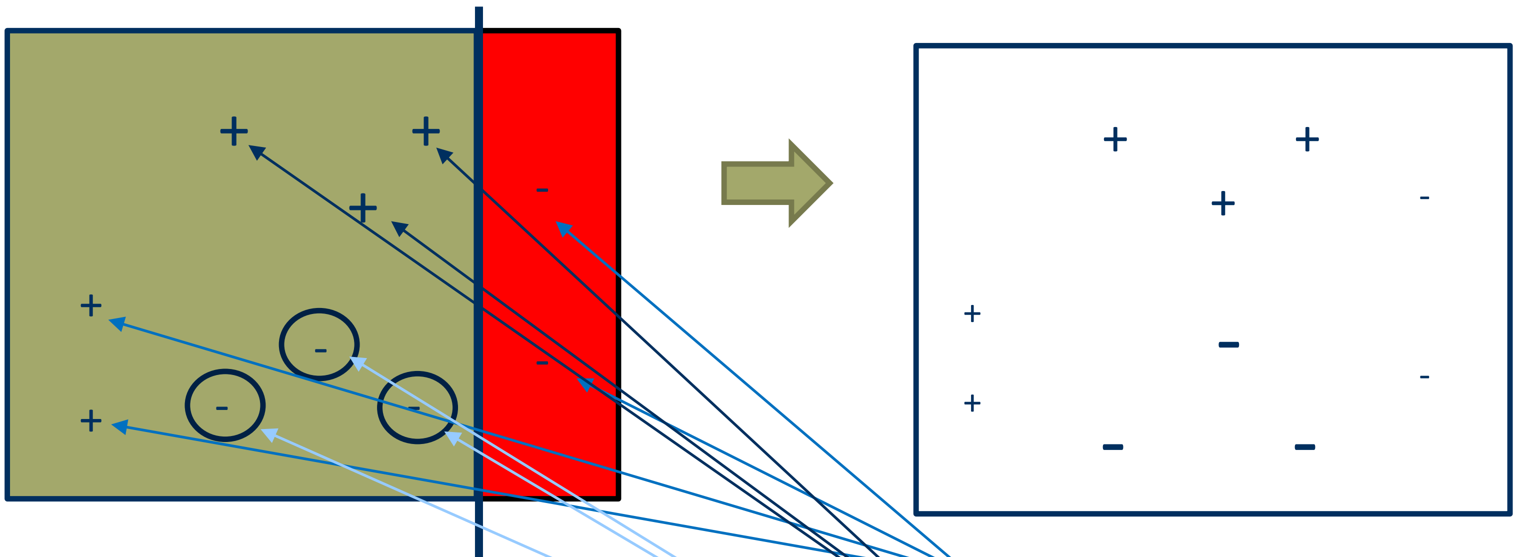
0.5/7 = **0.0714**

0.456/0.911 = 0.5

0.5/3 = **0.167**

Stockholm University

AdaBoost - Example



M2

$Err = 0.0714 * 3 = 0.21$

$\alpha_i = \frac{1}{2} \ln((1-Err)/Err) = \frac{1}{2} \ln((1-0.21)/0.21) = \mathbf{0.65}$

$w_j = w_j * e(-0.65) = 0.0714 * e(-0.65) = 0.37$

$w_j = w_j * e(-0.65) = 0.167 * e(-0.65) = 0.87$

$w_j = 0.0714 * e(0.65) = 1.37$

$0.37 * 4 = 1.48$

$0.87 * 3 = 2.61$

$1.37 * 3 = 4.11$

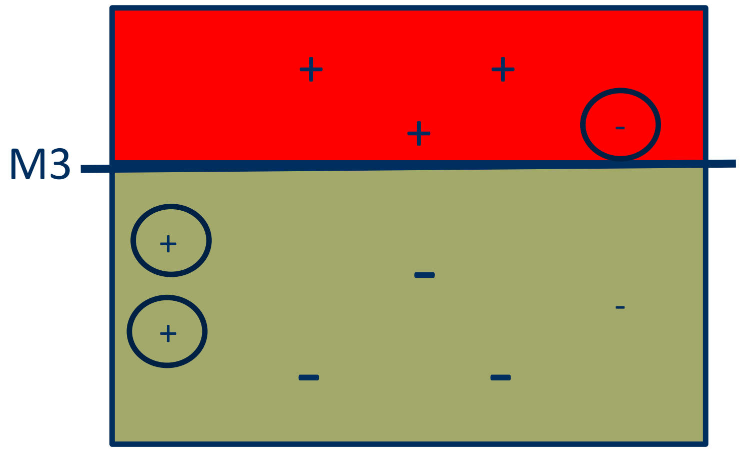
$1.48 + 2.61 + 4.11 = 8.2$

$1.48/8.2 = 0.18/4 = .045$

$2.61/8.2 = 0.32/3 = .106$

$4.11/8.2 = 0.5/3 = .166$

AdaBoost - Example

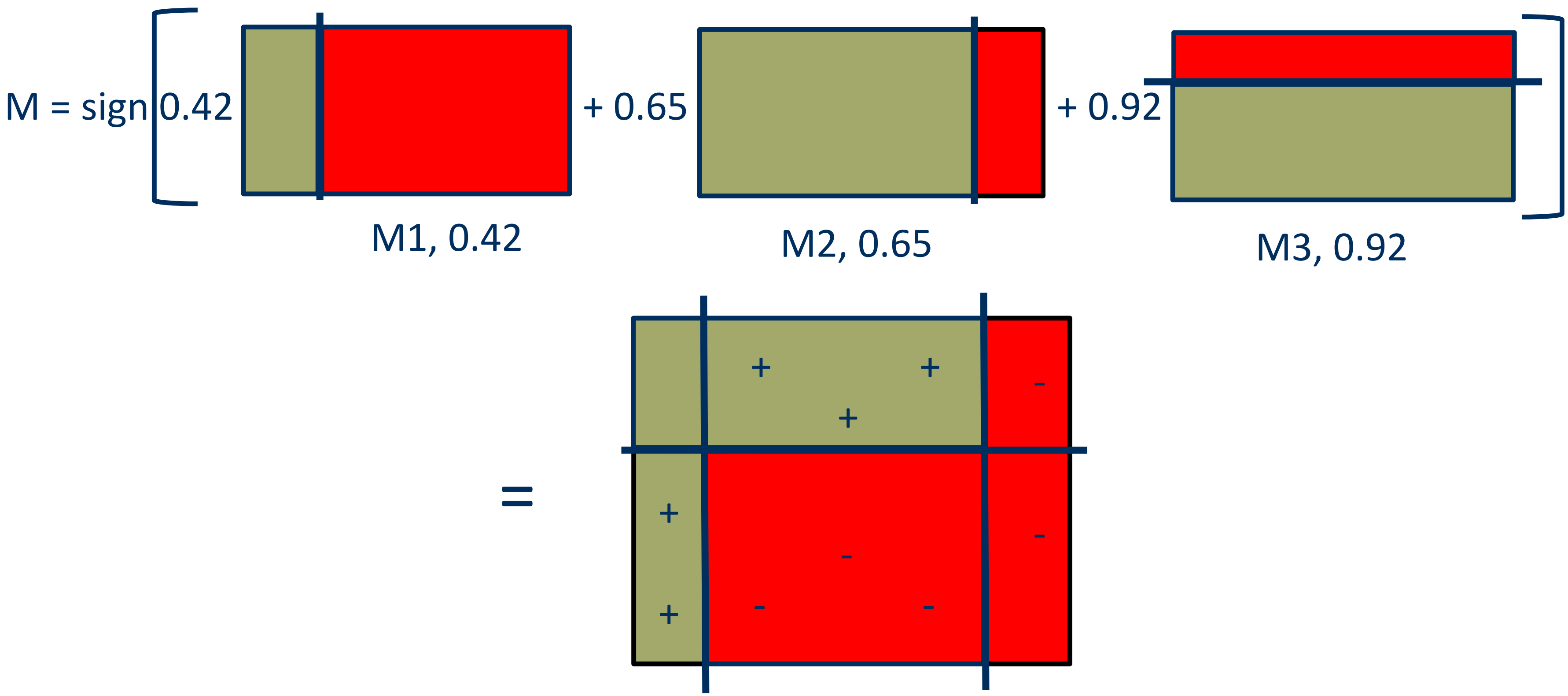


Err = 0.045 * 3 = **0.14**

αi = ½ ln((1-Err)/Err) = ½ ln((1-0.14)/0.14) = **0.92**

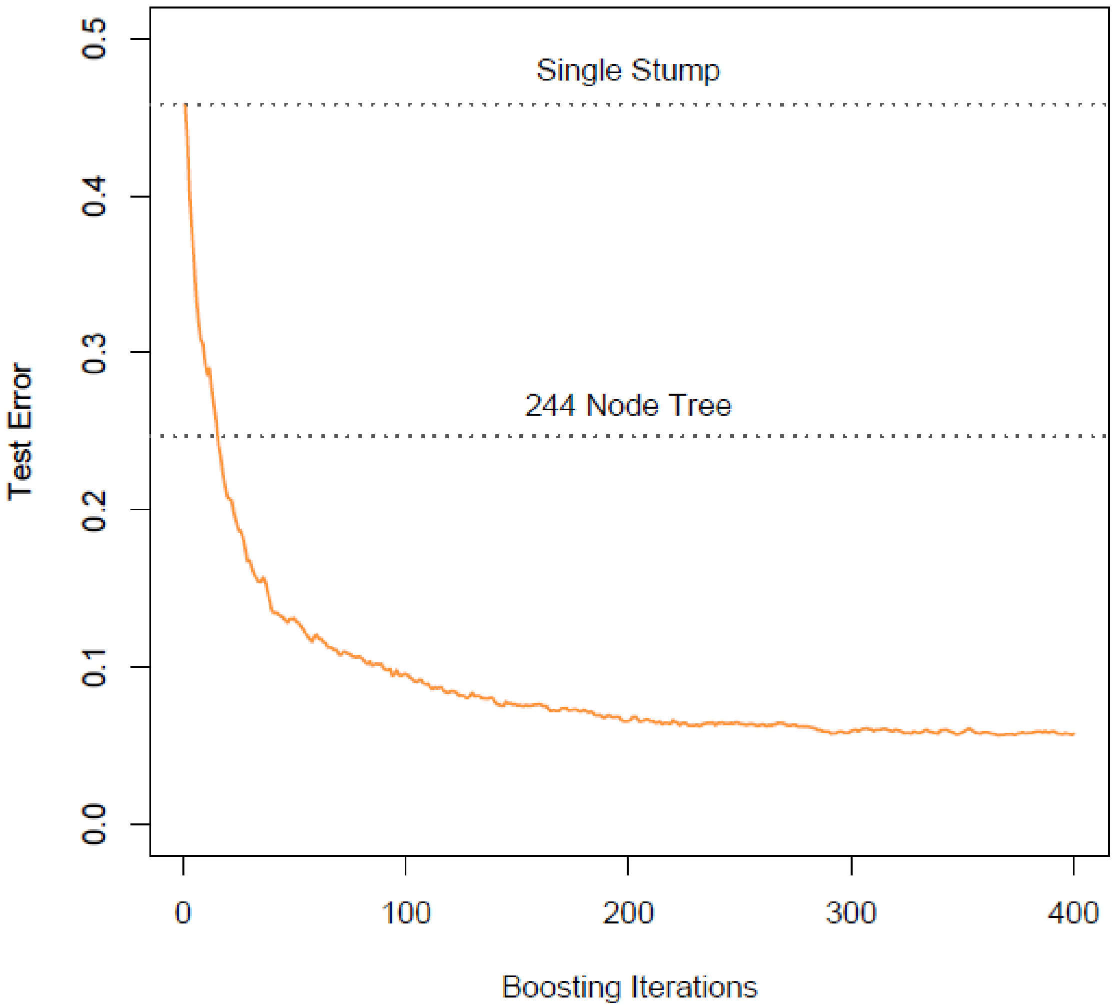
- Suppose we **stop** induction at round 3
- Then the ensemble consist of 3 predictors:
  - M1, M2, M3

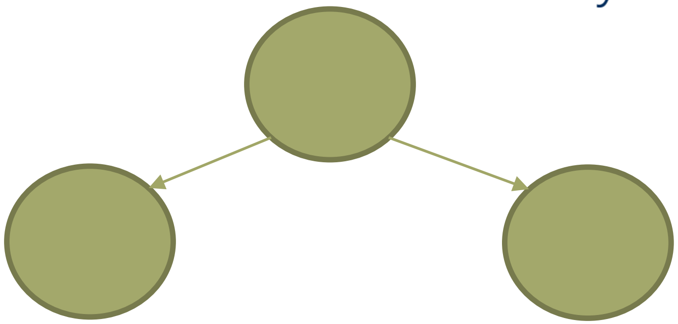
AdaBoost - Example



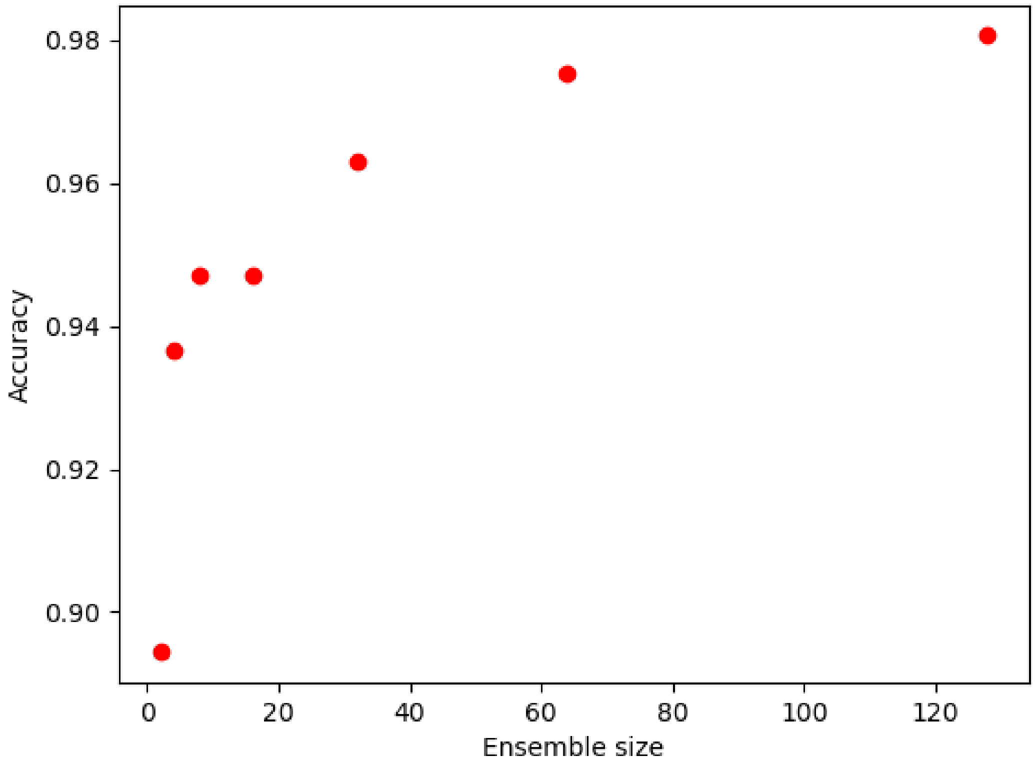
- Final predictor is a weighted linear combination of all predictors
- Where each predictor has weight αi
- Hence **multiple** weak, linear classifiers **combined** to give a strong nonlinear predictor

Decision stump





AdaBoost example result



Breast-cancer-Wisconsin dataset from UCI repository
Stratified 10 fold cross validation
Decision stump as base classifier

Gradient Boosting Machine - Regression

- Generalize Boosting methodology to use
  - Any **differentiable** loss functions
  - Gradient for mean squared error

Setting	*Loss function*	$-\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$

Input: instances (y1,x1), ..., (ym,xm), base learner BL, size s, learning rate l
Output: a sequence of models M0, ..., Ms

```
M0 = (y1 + ... + ym)/m                      #step 1 Initial model
for i = 1 to s:
  for j = 1 to m:
    yj = (yj - Mi-1(Xj))                    #step 2 Calculate residual values
    Mi = Mi-1 + l * BL({(y1,X1), ..., (ym,Xm)})  #step 3 Induce new tree + update Mi
```

Gradient Boosting Machine - Regression

Input: instances (y1,x1), ..., (ym,xm), base learner BL, size s, learning rate l

Output: a sequence of models M0, ..., Ms

```
M0 = (y1 + ... + ym)/m                          #step 1 Initial model
for i = 1 to s:
   for j = 1 to m:
      yj = (yj - Mi-1(Xj))                       #step 2 Calculate residual values
      Mi = Mi-1 + l * BL({(y1,X1), ..., (ym,Xm)}) #step 3 Induce new tree + update Mi
```

Weight	Hair	Length
65	Short	178
55	Long	160
78	Long	180
95	Short	193

Step 1 Initial model:

M0 = (y1 + ... + ym)/m

M0 = (178+160+180+193)/4 = 177.75

Step 2 Calculate residual values :

y1 = (y1 – M1-1(X1)) = (178 – 177.75) = 0.25

y2 = (y2 – M2-1(X1)) = (160 – 177.75) = -17.75

y3 = (180 – 177.75) = 2.25

y4 = (193 – 177.75) = 15.25

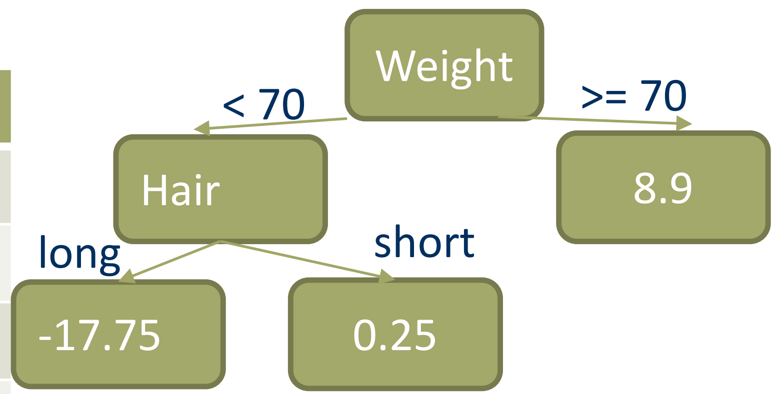
Gradient Boosting Machine - Regression

Stockholm University

```
Input: instances (y1,x1), ..., (ym,xm), base learner BL, size s, learning rate l
Output: a sequence of models M0, ..., Ms
M0 = (y1 + ... + ym)/m                        #step 1 Initial model
for i = 1 to s:
  for j = 1 to m:
    yj = (yj - Mi-1(Xj))                      #step 2 Calculate residual values
    Mi = Mi-1 + l * BL({(y1,X1), ..., (ym,Xm)})  #step 3 Induce new tree + update Mi
```

Step 3 Induce model M1:

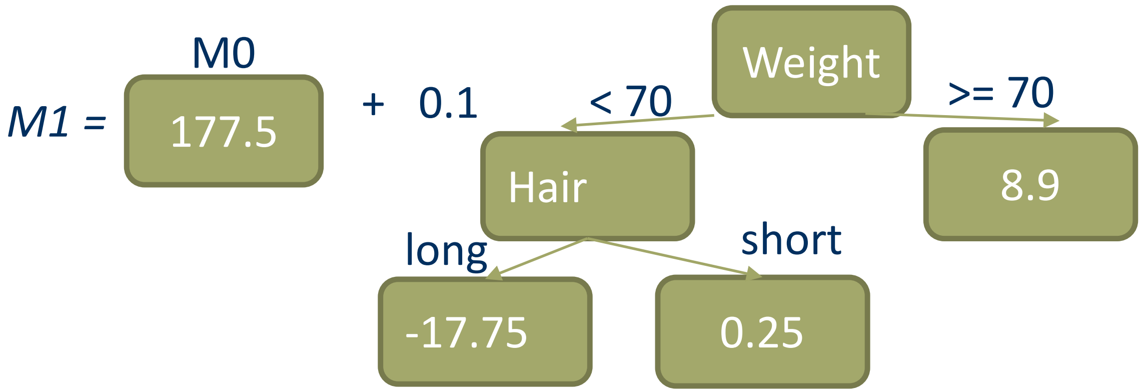
Weight	Hair	Length	Residual
65	Short	178	0.25
55	Long	160	-17.75
78	Long	180	2.25
95	Short	193	15.25



2.25 + 15.5 / 2 = 8.9

Gradient Boosting Machine - Regression

- Predict regression value on the training data (learning rate[0,1] = 0.1):



Weight	Hair	Length	Residual	N. Res
65	Short	178	0.25	0.2
55	Long	160	-17.75	-15.9
78	Long	180	2.25	1.6
95	Short	193	15.25	14.6

Calculate new resdiual values:

Y1 = 177.75 + 0.1 * 0.25 = 177.8

Y2 = 177.75 + 0.1 * -17.75 = 175.9

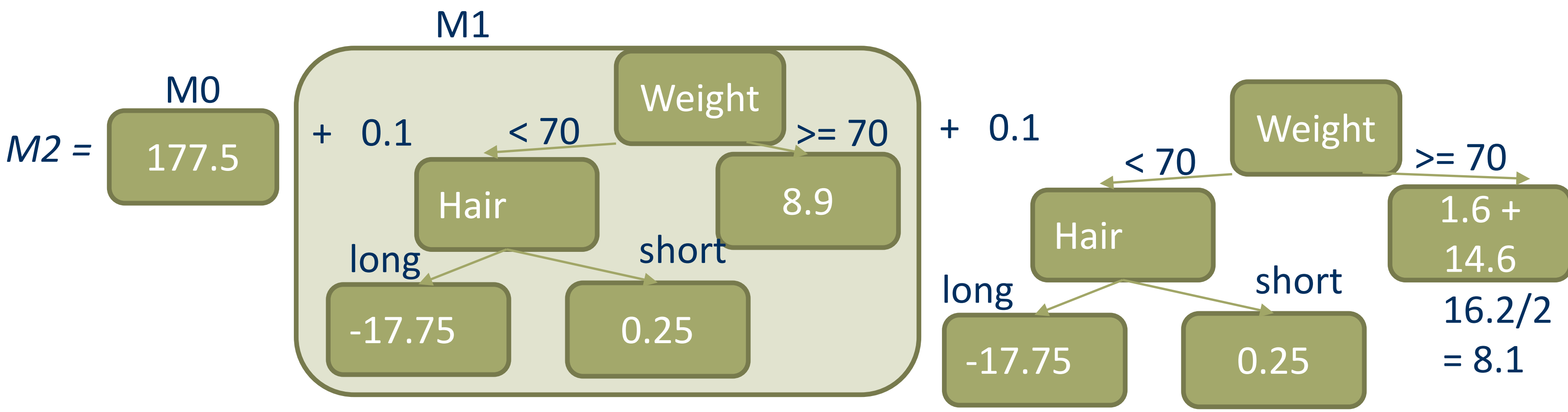
Y3 = 177.75 + 0.1 * 8.9 = 178.4

Y4 = 177.75 + 0.1 * 8.9 = 178.4

Gradient Boosting Machine - Regression

Stockholm University

Induce new tree:



Weight	Hair	Length	Residual	N. res.
65	Short	178	0.2	0.18
55	Long	160	-15.9	-15.5
78	Long	180	1.6	0.8
95	Short	193	14.6	13.8

Calculate new resdiual values:

Y1 = 177.8 + 0.1 * 0.2 = 177,82

Y2 = 175.9 + 0.1 * -15.9 = 174,3

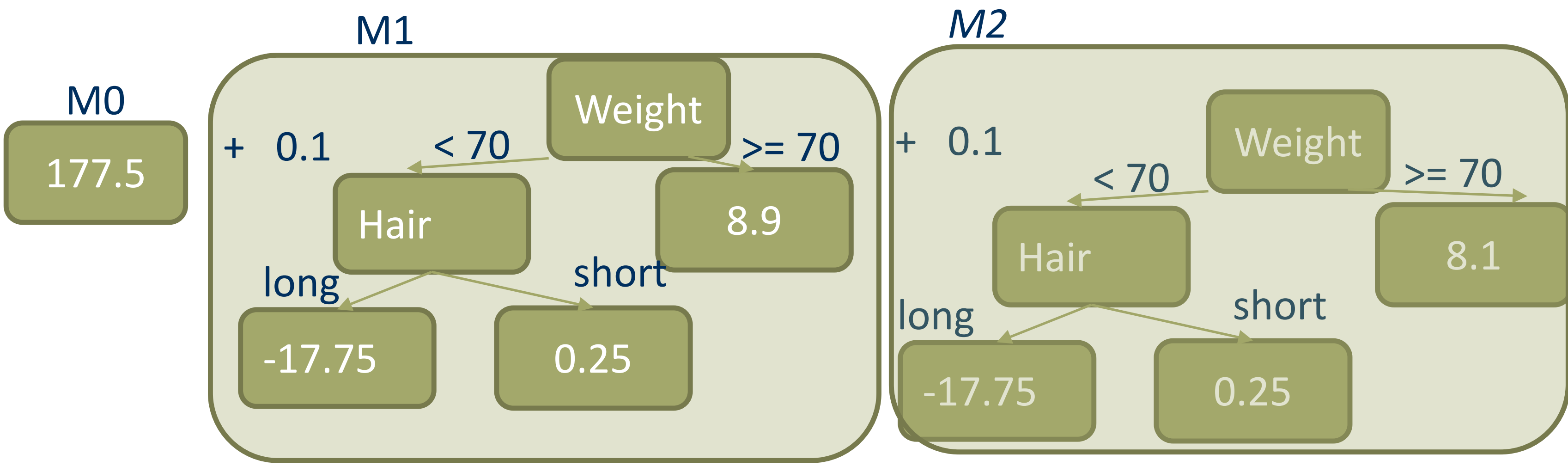
Y3 = 178.4 + 0.1 * 8.1 = 179,2

Y4 = 178.4 + 0.1 * 8.1 = 179,2

Gradient Boosting Machine - Regression

Stockholm University

Using the model:



Weight	Hair	Length
80	long	?

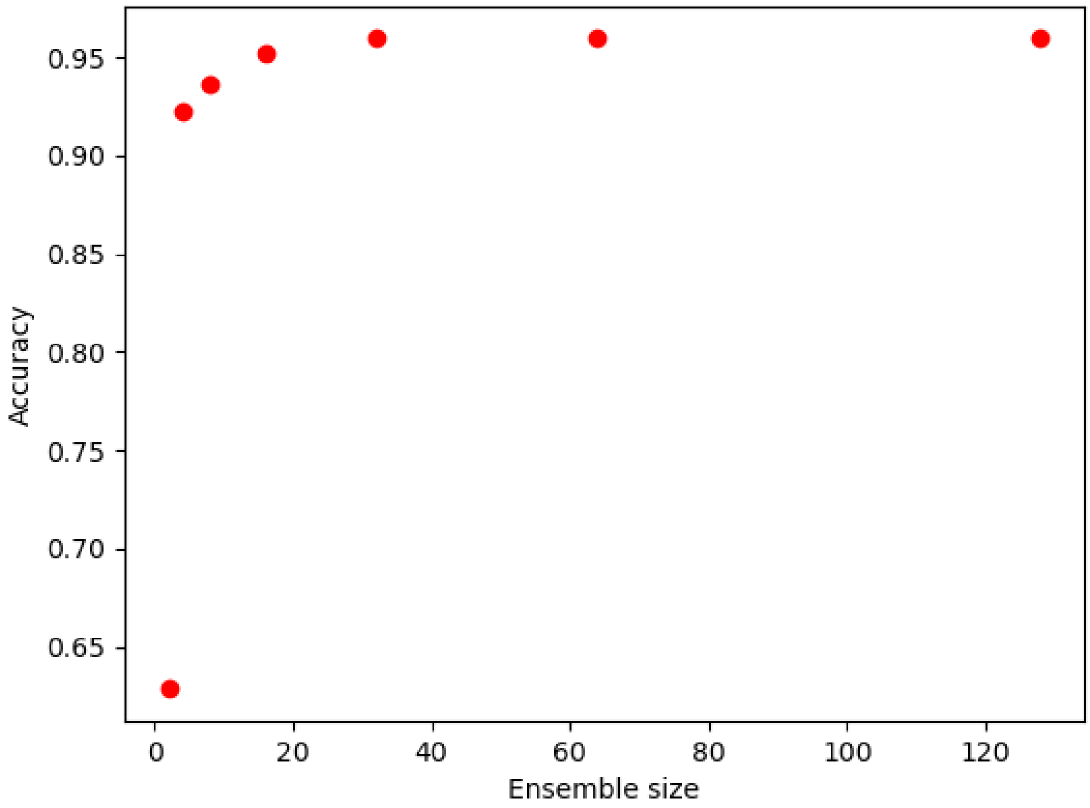
Predict = 177.75 + 0.1 * 8.9 + 0.1 * 8.1 = 179.45

Boosting tips

- For **multi-class** problems, it may be **difficult** for AdaBoost to reduce the error below 50% with weak base learners (like decision stumps)
  - More **powerful** base learners may be employed
  - The problem may be transformed into **multiple** binary classification problems (one against all)
  - Specific multi-class versions of AdaBoost have been developed

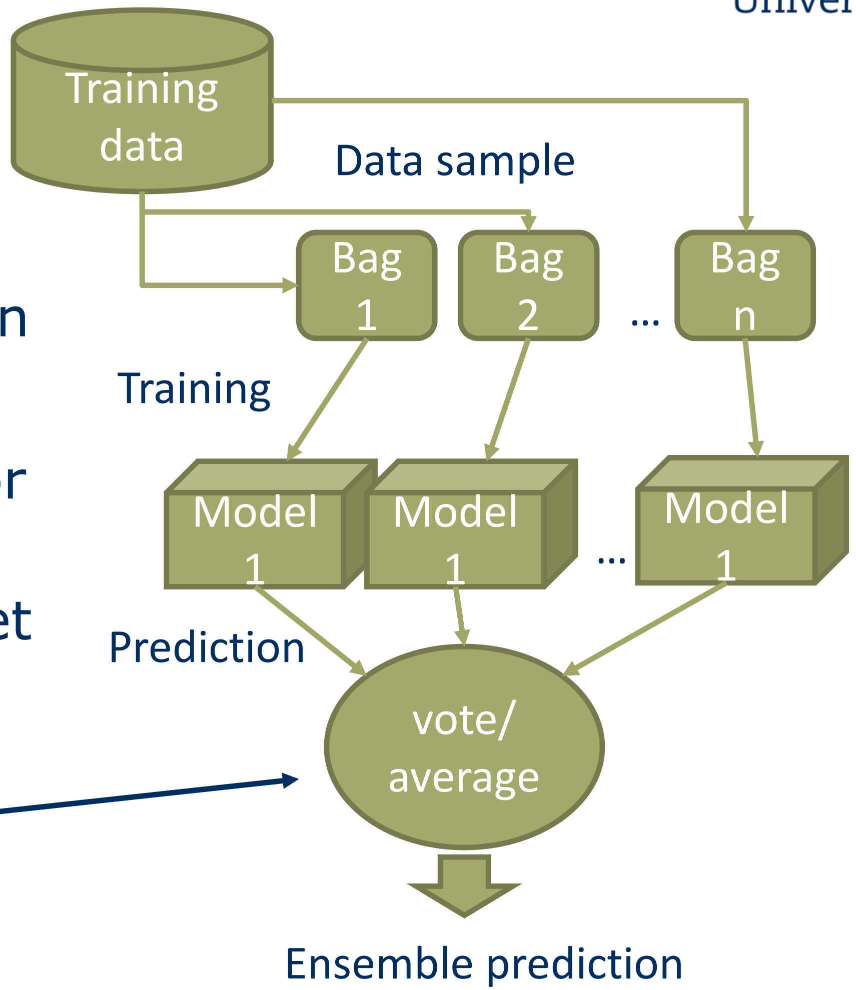
- Various **loss functions** may be used for the Gradient Boosting Machine (GBM), including log likelihood for classification
- GBM requires **tuning**
  - no. of iterations, model size and learning rate
- **XGBoost** implements GBM
  - well-known for winning many competitions, see (www.kaggle.com)
- Boosting can be **sensitive** to noise
  - erroneously labeled training examples

Gradient Boosting example result



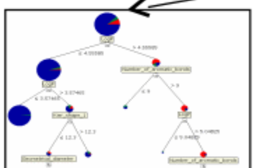
Stacking

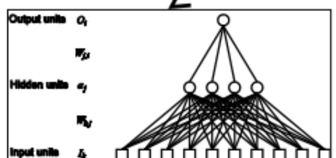
- Note that for example voting/averaging can be done in a weighted fashion
  - where weight come from for example
    - Accuracy on validation set
    - Or RMSError
- One can also use another **classifier** here
  - this is then called **stacking**

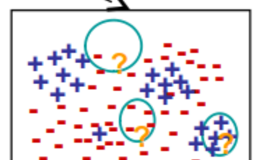


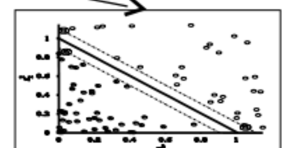
Stacking Example

Name	Solu-bility	Frac. of rotatable bonds	Geom diam.	Log P	Zagreb group index	Topol diam.	No. C atoms	No. heavy bonds
methylpentane	good	0.40	3.46	2.44	19	4	6	5
methylcyclohexene	good	0	3.00	2.51	31	4	7	7
nonene	med.	0.75	6.93	3.53	28	8	9	8
hexadiene	good	0.60	4.36	2.14	16	5	6	5
butadiene	good	0.33	2.65	1.36	8	3	4	3
naphthalene	good	0	3.61	2.84	57	5	10	11
acenaphthylene	good	0	3.58	3.32	83	5	12	14
pyrene	poor	0	5.00	4.58	117	7	16	19
dimethylanthracene	poor	0	5.29	4.61	108	7	16	18
hexahydropyrene	med.	0	5.00	3.82	117	7	16	19
triphenylene	poor	0	5.00	5.15	126	7	18	21
benzo(e)pyrene	poor	0	5.29	5.64	152	7	20	24









Class	M1	M2	M3	M4
good	good	good	good	med.
good	med.	good	good	good
med.	med.	good	med.	poor
poor	good	poor	good	good

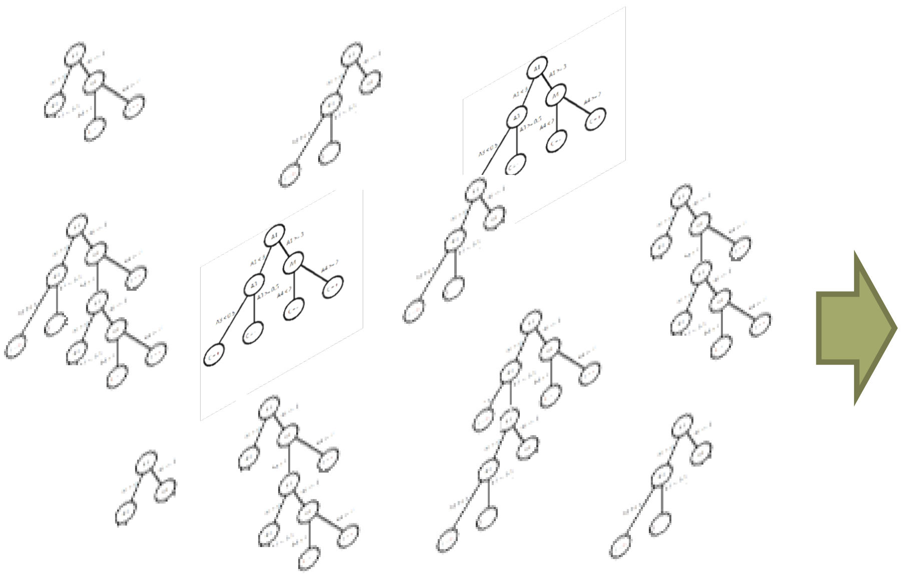
IF M2 = good & M3 = good THEN Class = good
IF M2 = poor THEN Class = poor
...

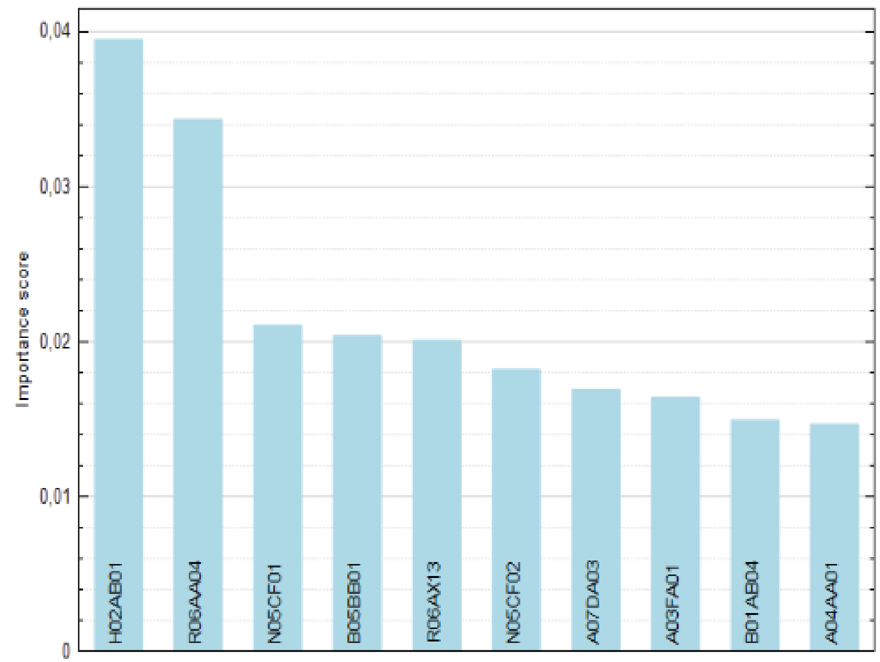
Stacking tips

- **Linear models** have been shown to be effective when learning the **combination** function
- Predictive performance can be improved by combining class **probability estimates** rather than **class labels** generated by the base models
- Why is this?

Stockholm University

Iterpretability of Ensembles





- There exist different approaches to estimating variable importance
  - one method is to measure the relative performance degradation (on OOB predictions) when permuting the values of each feature in turn.

Stockholm University

Summary

- Bias/Variance **tradeoff**
  - Ensemble methods that minimize **variance**
    - Bagging
    - Random Forests
  - Ensemble methods that minimize **bias**
    - AdaBoost
    - Gradient Boosting Machine

- Compared to the base learning algorithms, ensembles
  - typically substantially **improve** the predictive performance
    - sometimes to **state-of-the-art** performance

- The increased predictive **power** comes
  - at the cost of **reduced** interpretability

- The combination strategies may can lead to **increased** computational time (both in learning and prediction)
  - But this time penalty can be **eliminated** via parallelization

All for today

Questions?