Motivation

Different Predictors

- Neural Networks
- k nearest neighbors
- SVM
Motivation

Different Predictors

- Neural Networks
- k nearest neighbors
- SVM

**BUT:** All have their limitations. Overfitting (1-nearest neighbor), BlackBox (Neural Networks), How the hell does this work (SVM)...

Motivation

Different Predictors

- Neural Networks
- k nearest neighbors
- SVM

**BUT:** All have their limitations. Overfitting (1-nearest neighbor), BlackBox (Neural Networks), How the hell does this work (SVM)...

**So today:** We improve all of them!
Ensemble Learning

- (Iris’) Motivation: Wisdom of Crowds
- Basic Terms and Notations
- Overview of methods
- Bagging
- Boosting
- AdaBoost
Ensemble Learning consists of

1. Train many (weak) classifiers (or regression models)
2. Combine them to construct a classifier (regression model) more powerful than any of the individual ones
The Wisdom of Crowds
The collective knowledge of a *diverse* and *independent* body of people typically *exceeds* the knowledge of *any single individual* and can be harnessed by voting.

---


© Michael R. Berthold, Christian Borgelt, Frank Höppner, Frank Klawonn and Iris Adä
**The Wisdom of Crowds - Really?**

Crowd wiser than any individual

- When?
- For which questions?

See *The Wisdom of Crowds* by *James Surowiecki* published in 2004 to see this idea applied to business.
Consider this scenario

Ask each person in the crowd:

Will Mr. X win the general election in country Y?

The Crowds prediction:

MAJORITY answer.

→ This crowd predicts No. (Mr. X will not win the election.)
Has crowd made a good prediction?

If composition of crowd:
30% EXPERTS.
70% NON-EXPERTS.

and their level of expertise:
\[ P(\text{correct predict} | \text{expert}) = p_e \]
\[ P(\text{correct predict} | \text{non-expert}) = p_{ne} \]
Has crowd made a good prediction?

Let $p_e = .8$ and $p_{ne} = .5$

For a random person from the crowd
$P(\text{correct predict}|\text{individual}) = .3p_e + .7p_{ne} = .59$
Has crowd made a good prediction?

Let $p_e = .8$ and $p_{ne} = .5$

$P(\text{correct predict} | \text{individual}) = p_i = .59$

**If crowd contains 50 independent people:**

$P(\text{correct predict} | \text{crowd}) = \sum_{k=26}^{50} \binom{50}{k} p_i^k (1 - p_i)^{50-k}$

$= 0.8745$
Has crowd made a good prediction?

Let $p_e = .8$ and $p_{ne} = .5$

$P(\text{correct predict} | \text{individual}) = p_i = .59$

If crowd contains 50 independent people:

$P(\text{correct predict} | \text{crowd}) = \sum_{k=26}^{50} \binom{50}{k} p_i^k (1 - p_i)^{50-k}$

$= 0.8745$

This crowd has made a prediction with probability $0.875$ of being correct which is $> p_e$. It is wiser than each of the experts!
But...

Why didn't I just asked a bunch of experts??

- Large enough crowd $\rightarrow$ high probability a sufficient number of experts will be in crowd (for any question).
- Random selection $\rightarrow$ don’t make a biased choice in experts.
- For some questions it may be hard to identify a diverse set of experts.
But...

Why didn’t I just asked a bunch of experts??

- Large enough crowd $\rightarrow$ high probability a sufficient number of experts will be in crowd (for any question).
- **Random selection $\rightarrow$ don’t make a biased choice in experts.**
- For some questions it may be hard to identify a diverse set of experts.
But...

Why didn't I just asked a bunch of experts??

- Large enough crowd $\rightarrow$ high probability a sufficient number of experts will be in crowd (for any question).
- Random selection $\rightarrow$ don’t make a biased choice in experts.
- For some questions it may be hard to identify a diverse set of experts
For a random crowd

Given a random question expect each person to have a different level of expertise.

Will it rain tomorrow?

← redness proportional to expertise
Given a **random question** expect each **person** to have a **different level** of **expertise**.

Will the world go down in 2012?

← redness proportional to expertise
What makes a crowd wise?

According to James Surowiecki there are four elements required to form a wise crowd:

- **Diversity of opinion.** People in crowd should have a range of experiences, education and opinions. (Encourages independent predictions)
- **Independence.** Prediction by person in crowd is not influenced by other people in the crowd.
- **Decentralization.** People have specializations and local knowledge.
- **Aggregation.** There is a mechanism for aggregating all predictions into one single prediction.
What makes a crowd wise?

According to James Surowiecki there are four elements required to form a wise crowd

- **Diversity of opinion.** People in crowd should have a range of experiences, education and opinions. (Encourages independent predictions)

- **Independence.** Prediction by person in crowd is not influenced by other people in the crowd.

- **Decentralization.** People have specializations and local knowledge.

- **Aggregation.** There is a mechanism for aggregating all predictions into one single prediction.
What makes a crowd wise?

According to *James Surowiecki* there are four elements required to form a wise crowd:

- **Diversity of opinion.** People in crowd should have a range of experiences, education and opinions. (Encourages independent predictions)

- **Independence.** Prediction by person in crowd is not influenced by other people in the crowd.

- **Decentralization.** People have specializations and local knowledge.

- **Aggregation.** There is a mechanism for aggregating all predictions into one single prediction.
What makes a crowd wise?

According to James Surowiecki there are four elements required to form a wise crowd

- **Diversity of opinion.** People in crowd should have a range of experiences, education and opinions. (Encourages independent predictions)
- **Independence.** Prediction by person in crowd is not influenced by other people in the crowd.
- **Decentralization.** People have specializations and local knowledge.
- **Aggregation.** There is a mechanism for aggregating all predictions into one single prediction.
The crowd must be careful

In the analysis of the crowd it is implicitly assumed:
- each person is not concerned with the opinions of others,
- no-one is copying anyone else in the crowd.

In the analysis of the crowd we implicitly assumed:
- The non-experts will predict a **completely random wrong answer** - these will cancel each other out (to some degree).
- However, there may be a systematic and consistent bias in the non-experts predictions.
Back to machine learning
We will exploit **Wisdom of crowd** ideas for specific tasks by
- combining (classifier) predictions and
- aim to combine independent and diverse predictors (classifiers).

We can also use labeled training data
- to identify the expert classifiers in the pool;
- to identify complementary classifiers;
- to indicate how to best combine them.
Basic Terms

- Remember? Bias = model error + algorithmic error
  - Model error: the error we get by selecting a model
  - Algorithmic error: by selecting the algorithm itself and the parameters of the algorithm
- Base-Learning: Fixed Bias / User parameterized
- Meta-Learning: Dynamic bias selection using meta knowledge
- Meta-Knowledge: Knowledge achieved during the learning process
Ensemble Learning for Algorithm Recommendation
## Combining base-learners: Categories

<table>
<thead>
<tr>
<th>Philosophy</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagging, Boosting</td>
<td>Variation in data</td>
</tr>
<tr>
<td>Stacking</td>
<td>Variation among learners (multi-expert)</td>
</tr>
<tr>
<td>Cascading, Delegating</td>
<td>Variation among learners (multi-stage)</td>
</tr>
<tr>
<td>Arbitrating</td>
<td>Variation among learners (refereed)</td>
</tr>
<tr>
<td>Meta decision trees</td>
<td>Variation in data and among learners</td>
</tr>
</tbody>
</table>
Bagging and Boosting
Bagging and Boosting

- Best-known techniques
- Based on selection of multiple data sub sets
- Meta model is created by combining the base models

Advantages:
- Reduces overfitting
- Most effective when the base learner is highly sensitive to data
- Typically increases accuracy

Disadvantages:
- Interpretability of interpretable base learners is lost
Can a set of weak learners create a single strong learner?

**Bagging:**
- Select $N$ independent samples of the Training Data
- Learn one model on each of the samples $\rightarrow h_1, \ldots, h_N$
- Classification: Use the class most predicted by all classifiers
- Regression: Use the mean of all predictions

**Boosting:**
- Tries to learn a weighting for the models
- Later weak learners focus more on the examples that previous weak learners misclassified
- There is no single "best" boosting method
Can a set of weak learners create a single strong learner?

**Bagging:**
- **Select** $N$ independent samples of the Training Data
- Learn one model on each of the samples $\rightarrow h_1, ..., h_N$

  - Classification: Use the class most predicted by all classifiers
  - Regression: Use the mean of all predictions

**Boosting:**
- Tries to learn a weighting for the models
- Later weak learners focus more on the examples that previous weak learners misclassified
- There is no single "best" boosting method
Can a set of weak learners create a single strong learner?

- **Bagging**:
  - Select $N$ independent samples of the Training Data
  - **Learn one model on each of the samples** $\rightarrow h_1, \ldots, h_N$
  - Classification: Use the class most predicted by all classifiers
  - Regression: Use the mean of all prediction

- **Boosting**:
  - Tries to learn a weighting for the models
  - Later weak learners focus more on the examples that previous weak learners misclassified
  - There is no single "best" boosting method
Can a set of weak learners create a single strong learner?

- **Bagging:**
  - Select $N$ independent samples of the Training Data
  - Learn one model on each of the samples \(\rightarrow h_1, \ldots, h_N\)

- **Classification:** Use the class most predicted by all classifiers
- **Regression:** Use the mean of all predictions

- **Boosting:**
  - Tries to learn a weighting for the models
  - Later weak learners focus more on the examples that previous weak learners misclassified
  - There is no single "best" boosting method
Can a set of weak learners create a single strong learner?

- **Bagging:**
  - Select $N$ independent samples of the Training Data
  - Learn one model on each of the samples $\rightarrow h_1, \ldots, h_N$

  - Classification: Use the class most predicted by all classifiers
  - Regression: Use the mean of all prediction

- **Boosting:**
  - Tries to learn a weighting for the models
  - Later weak learners focus more on the examples that previous weak learners misclassified
  - There is no single "best" boosting method
Can a set of weak learners create a single strong learner?

- **Bagging:**
  - Select $N$ independent samples of the Training Data
  - Learn one model on each of the samples $\rightarrow h_1, \ldots, h_N$
  - Classification: Use the class most predicted by all classifiers
  - Regression: Use the mean of all predictions

- **Boosting:**
  - Tries to learn a weighting for the models
  - Later weak learners focus more on the examples that previous weak learners misclassified
  - There is no single "best" boosting method
Can a set of weak learners create a single strong learner?

- **Bagging**:
  - Select $N$ independent samples of the Training Data
  - Learn one model on each of the samples $\rightarrow h_1, \ldots, h_N$

- Classification: Use the class most predicted by all classifiers
- Regression: Use the mean of all prediction

- **Boosting**:
  - **Tries to learn a weighting for the models**
  - Later weak learners focus more on the examples that previous weak learners misclassified
  - There is no single "best" boosting method
Can a set of weak learners create a single strong learner?

- **Bagging**: 
  - Select $N$ independent samples of the Training Data 
  - Learn one model on each of the samples $\rightarrow h_1, \ldots, h_N$

  - Classification: Use the class most predicted by all classifiers
  - Regression: Use the mean of all predictions

- **Boosting**: 
  - Tries to learn a weighting for the models 
  - Later weak learners focus more on the examples that previous weak learners misclassified
  - There is no single "best" boosting method
Can a set of weak learners create a single strong learner?

- **Bagging**:
  - Select $N$ independent samples of the Training Data
  - Learn one model on each of the samples $\rightarrow h_1, \ldots, h_N$
  - Classification: Use the class most predicted by all classifiers
  - Regression: Use the mean of all prediction

- **Boosting**:
  - Tries to learn a weighting for the models
  - Later weak learners focus more on the examples that previous weak learners misclassified
  - **There is no single "best" boosting method**
Boosting

One boosting method after Schapire

**Training**:
- Create $c_1$: weak learner on a sample $t_1$ of the data
- Create $t_2$: sample which is 50% miss classified by $c_1$
- Create $c_2$: weak learner on the sample $t_2$
- Create $t_3$: subset of the data where $c_1$ predicts differently than $c_2$
- Create $c_3$: weak learner on the sample $t_3$

**Classification**:
- Classify with $c_1$ and $c_2$
- If unequal, use $c_3$ as final classification
Stacking and Cascade Generalization
Stacking

- In Bagging and Boosting: we used always the same base learner

- **Stacking uses differences among learners**

- **Two levels of learning**
  1. Base learners are trained, each on the whole data set
  2. Meta learners are created on meta data (e.g. predicted class) obtained in level 1

- **Two levels of classifying**
  1. Base learner are used on data point
  2. Meta learners are applied on base learner predictions
Stacking

- In Bagging and Boosting: we used always the same base learner

- Stacking uses differences among learners

- **Two levels of learning**
  1. Base learners are trained, each on the whole data set
  2. Meta learners are created on meta data (e.g. predicted class) obtained in level 1

- **Two levels of classifying**
  1. Base learner are used on data point
  2. Meta learners are applied on base learner predictions
Stacking

- In Bagging and Boosting: we used always the same base learner

- Stacking uses differences among learners

- Two levels of learning
  1. **Base learners are trained, each on the whole data set**
  2. Meta learners are created on meta data (e.g. predicted class) obtained in level 1

- Two levels of classifying
  1. Base learner are used on data point
  2. Meta learners are applied on base learner predictions
Stacking

- In Bagging and Boosting: we used always the same base learner

- Stacking uses differences among learners

- Two levels of learning
  1. Base learners are trained, each on the whole data set
  2. **Meta learners are created on meta data (e.g. predicted class) obtained in level 1**

- Two levels of classifying
  1. Base learner are used on data point
  2. Meta learners are applied on base learner predictions
Stacking

- In Bagging and Boosting: we used always the same base learner

- Stacking uses differences among learners
- Two levels of learning
  1. Base learners are trained, each on the whole data set
  2. Meta learners are created on meta data (e.g. predicted class) obtained in level 1
- Two levels of classifying
  1. Base learners are used on data point
  2. Meta learners are applied on base learner predictions
Stacking

- In Bagging and Boosting: we used always the same base learner

- Stacking uses differences among learners

- Two levels of learning
  1. Base learners are trained, each on the whole data set
  2. Meta learners are created on meta data (e.g. predicted class) obtained in level 1

- Two levels of classifying
  1. **Base learner are used on data point**
  2. Meta learners are applied on base learner predictions
Stacking

- In Bagging and Boosting: we used always the same base learner

- Stacking uses differences among learners

- Two levels of learning
  1. Base learners are trained, each on the whole data set
  2. Meta learners are created on meta data (e.g. predicted class) obtained in level 1

- Two levels of classifying
  1. Base learner are used on data point
  2. **Meta learners are applied on base learner predictions**
Cascade Generalization

Stacking: Base learners are used parallel

- Here: Base learners are used in a sequence with "partial" meta learners
- Knowledge from previous classifiers can be used in later ones
- After each base learner the data set is adjusted using the new information

For classification:

- Only the last model is used
- Which incorporates the knowledge of previous models (all base methods are used)
Cascading and Delegating
Cascading and Delegating

- Until now: all base classifiers are used for classification
- Here: Multistage classifiers, not all are required for classification
- Main advantage: faster classification
Cascading

- Multilearner version of boosting
- Uses learned confidence of previous models
- Train base learner $h_i$ using knowledge from previous base learner...
- ...on data, which was most probably misclassified by previous learners
- **Classify:** go through all base models, stop and use as classification if the model has a confidence greater than epsilon
Delegating

- Cascading: all instances are used in each step
- Delegating: only instances below confidence threshold are processed in the next step
- Idea:
  - Use everything and test for which data points you are good enough
  - Pass the remaining work to someone else.
  - If there is no someone else, ... guess

Advantages:
- Still understandable (no model combination)
- Improved efficiency, due to the decreasing number of examples.
Ada Boost

Some properties

- Freund & Schapire (1995)
Ada Boost

Some properties

- Freund & Schapire (1995)
- AB is a linear classification algorithm
Some properties

- Freund & Schapire (1995)
- AB is a linear classification algorithm
- AB has good generalization properties (Avoids overfitting as long as the training data is not too noisy)
Ada Boost

Some properties

- Freund & Schapire (1995)
- AB is a linear classification algorithm
- AB has good generalization properties (Avoids overfitting as long as the training data is not too noisy)
- AB is a feature selector
Ada Boost

Example

Weak classifier $h_1$

Weak classifier $h_2$

Weak classifier $h_3$

Strong classifier $= (\alpha_1 h_1) + (\alpha_2 h_2) + \ldots + (\alpha_T h_T)$
Terminology

- **Strong classifier**: \( \sum_{t=1}^{T} (\alpha_t h_t(x)) \)
- Where \( h_t(x) \) is a weak classifier weighted by \( \alpha_t \)
- Classification result: \( H(x) = \text{sgn}\left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \)

Comment: \( h_t(x) \)'s can be thought of as simple features
**Algorithm** Ada Boost

Initialize weight of $x_i$ with $D_0(i) = \frac{1}{m}$

**for** $t = 1, \ldots, T$: **do**

1. $h_t$ = “Weak-Learner”
2. Calculate error $\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(y_i, h_t(x_i))$
3. Calculate weight of learner $\alpha_t = \log \frac{1-\epsilon_t}{\epsilon_t}$
4. Update the weights $D_t(i)$ of all $x_i$

**end for**

Resulting classifier

\[ H(x) = \text{sgn} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right) \]
Ada Boost - Pseudo code - Overview

Algorithm Ada Boost

1. Initialize weight of $x_i$ with $D_0(i) = \frac{1}{m}$
2. for $t = 1, ..., T$: do
   1. $h_t = \text{"Weak-Learner"}$
   2. Calculate error $\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(y_i, h_t(x_i))$
   3. Calculate weight of learner $\alpha_t = \log \frac{1 - \epsilon_t}{\epsilon_t}$
   4. Update the weights $D_t(i)$ of all $x_i$
3. end for

Resulting classifier

$H(x) = \text{sgn} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$
Ada Boost - Weak Learners

**Algorithm**  Weak learner

Train a set $H$ of many many weak-learners $h$ on the data
Return the one $h$ with the lowest weighted classification error

$$\epsilon = \arg \min_{h_j \in H} (\sum_{i=1}^{m} D(i) \delta(y_i, h_t(x_i)))$$

Make sure : $\epsilon < 0.5$
Algorithm Ada Boost

Initialize weight of $x_i$ with $D_0(i) = \frac{1}{m}$

for $t = 1, \ldots, T$: do
    1. $h_t$ = “Weak-Learners”
    2. Calculate error $\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(y_i, h_t(x_i))$
    3. Calculate weight of learner $\alpha_t = \log \frac{1-\epsilon_t}{\epsilon_t}$
    4. Update the weights $D_t(i)$ of all $x_i$

end for

Resulting classifier

$H(x) = \text{sgn} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$
Algorithm Ada Boost

Initialize weight of $x_i$ with $D_0(i) = \frac{1}{m}$

for $t = 1, ..., T$: do

1. $h_t = “Weak-Learners”$
2. Calculate error $\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(y_i,h_t(x_i))$
3. Calculate weight of learner $\alpha_t = \log \frac{1-\epsilon_t}{\epsilon_t}$
4. Update the weights $D_t(i)$ of all $x_i$

end for

Resulting classifier

$H(x) = \text{sgn} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$
Algorithm Ada Boost

- Initialize weight of $x_i$ with $D_0(i) = \frac{1}{m}$
- for $t = 1, ..., T$: do
  1. $h_t = \text{"Weak-Learners"}$
  2. Calculate error $\epsilon_t = \sum_{i=1}^{m} D_t(i) \delta(y_i, h_t(x_i))$
  3. Calculate weight of learner $\alpha_t = \log \frac{1-\epsilon_t}{\epsilon_t}$
  4. Update the weights $D_t(i)$ of all $x_i$
- end for
- Resulting classifier

$$H(x) = \text{sgn} \left( \sum_{t=1}^{T} \alpha_t h_t(x) \right)$$
Update of the weights of the training samples

Update weights an normalize them

\[
D_{t+1}(i) = \frac{D_t(i)}{Z_t} \exp(-\alpha_t y_i h_t(x_i))
\]

\[
Z_t = \sum_{i=0}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))
\]
AdaBoost

Update of the weights of the training samples

Update weights an normalize them

\[
D_{t+1}(i) = \frac{D_t(i)}{Z_t} \exp\left(-\alpha_t y_i h_t(x_i)\right)
\]

\[
Z_t = \sum_{i=0}^{m} D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)
\]

Comment

- Weight of correct classified examples is decreased
- Weight of incorrect classified examples is increased

\[
\exp(-\alpha y h(x_i)) = \begin{cases} 
< 1 & y = h(x_i) \\
> 1 & y \neq h(x_i)
\end{cases}
\]
Weighting the weak-learners “Upper-Bound Theorem”

- Primary goal is to minimize

\[
\epsilon_{tr}(H) = \frac{1}{m} |\{i : H(x_i) \neq y_i\}|
\]

- Global error is bounded by

\[
\epsilon_{tr} \leq \sum_{t=1}^{T} Z_t
\]

\[
Z_t = \sum_{i=0}^{m} D_t(i) \exp (-\alpha_t y_i h_t(x_i))
\]
AdaBoost - Why does it work?

Weighting the weak-learners

That’s why
Weighting the weak-learners

That’s why

- Minimizing $Z_t = \sum_{i=0}^{m} D_t(i) \exp (-\alpha_t y_i h_t(x_i))$ results in a minimization of the global error
Weighting the weak-learners

That’s why

- Minimizing $Z_t = \sum_{i=0}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$ results in a minimization of the global error.
- Upper-Bound can be minimized by ...
  1. Choosing the optimal hypothesis $h_t$ ...
  2. ... with an optimal weight $\alpha_t$
AdaBoost - Why does it work?

Weighting the weak-learners

That’s why

- Minimizing $Z_t = \sum_{i=0}^{m} D_t(i) \exp (-\alpha_t y_i h_t(x_i))$ results in a minimization of the global error
- Upper-Bound can be minimized by ...
  1. Choosing the optimal hypothesis $h_t$ ...
  2. ... with an optimal weight $\alpha_t$
- Minimizing $Z_t$ results in $\alpha_t = \log \frac{1-\epsilon_t}{\epsilon_t}$
AdaBoost

Advantages

- Very simple to implement
- Feature selection on very large features spaces
- Fairly good generalization

Disadvantages

- Can overfit in presence of noise
- Unclear which weak-learning algorithm fits best for a given problem
**Advantages**
- Very simple to implement
- Feature selection on very large features spaces
- Fairly good generalization

**Disadvantages**
- Can overfit in presence of noise
- Unclear which weak-learning algorithm fits best for a given problem
Recap

Ensemble Learning

- ... combines multiple base learners into one powerful meta learner
Recap

Ensemble Learning

- ... combines multiple base learners into one powerful meta learner
- ... can be used as a tool for algorithm recommendation
Ensemble Learning

- ... combines multiple base learners into one powerful meta learner
- ... can be used as a tool for algorithm recommendation
- ... can be used to guess algorithms parameters
Ensemble Learning

- ... combines multiple base learners into one powerful meta learner
- ... can be used as a tool for algorithm recommendation
- ... can be used to guess algorithms parameters
- ... various methods exist:
Recap

Ensemble Learning

- ... combines multiple base learners into one powerful meta learner
- ... can be used as a tool for algorithm recommendation
- ... can be used to guess algorithms parameters
- ... various methods exist:
  - Bagging and Boosting
Recap

Ensemble Learning

- ... combines multiple base learners into one powerful meta learner
- ... can be used as a tool for algorithm recommendation
- ... can be used to guess algorithms parameters
- ... various methods exist:
  - Bagging and Boosting
  - Stacking and Cascade Generalization
Recap

Ensemble Learning

- ... combines multiple base learners into one powerful meta learner
- ... can be used as a tool for algorithm recommendation
- ... can be used to guess algorithms parameters
- ... various methods exist:
  - Bagging and Boosting
  - Stacking and Cascade Generalization
  - **Cascading and Delegating**
Ensemble Learning

- combines multiple base learners into one powerful meta learner
- can be used as a tool for algorithm recommendation
- can be used to guess algorithms parameters
- various methods exist:
  - Bagging and Boosting
  - Stacking and Cascade Generalization
  - Cascading and Delegating
  - and more