Ensemble Methods

TECHNISCHE UNIVERSITÄT DARMSTADT

Data Mining and Machine Learning: Techniques and Algorithms

Eneldo Loza Mencía

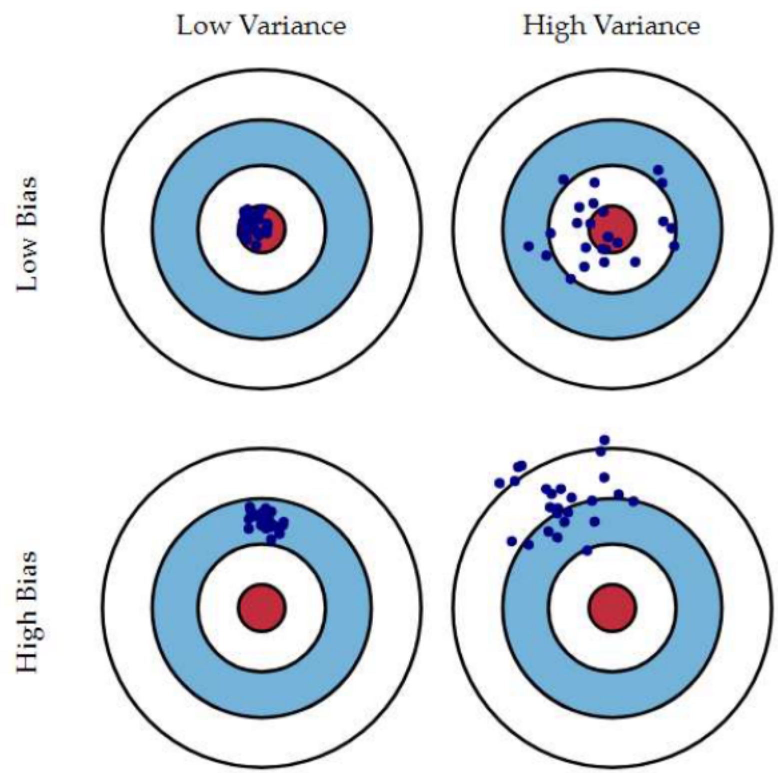
eneldo@ke.tu-darmstadt.de

Knowledge Engineering Group, TU Darmstadt

International Week 2019, 21.1. – 24.1.
University of Economics, Prague

Bias and Variance Decomposition

- Bias:
  - the part of the error that is caused by bad (non-appriopriate!) model
- Variance:
  - the part of the error that is caused by the data sample



Theoretical interpretation

- suppose a regression problem $f(x)$ and we measure the error by squared loss by learning different models $\bar{f}(x)$
- we obtain the expected loss

$$\mathbb{E}\left[\left(f(x)-\hat{f}(x)\right)^2\right] = \underbrace{\left(f(x)-\mathbb{E}\left[\hat{f}(x)\right]\right)^2}_{\text{bias: systematic error of all } \bar{f}(x)} + \underbrace{\mathbb{E}\left[\left(\hat{f}(x)-\mathbb{E}\left[\hat{f}(x)\right]\right)^2\right]}_{\text{variance: variability of } \bar{f}(x)}$$

Peter Flach: Machine Learning The Art and Science of Algorithms that Make Sense of Data
Slides CS6375: Machine Learning

Bias and Variance Decomposition
Bias-Variance Trade-off/Dilemma

- Models with a low bias often have a high variance
  - small variations in training data may result in considerably different models
  - often powerful models with high number of parameters
  - e.g., nearest neighbor, unpruned decision trees, SVM with kernels, big neural networks
- Models with a low variance often have a high bias
  - suffer less from variability due to random variations in training data
  - but may introduce systematic bias that large amount of training data cannot solve
  - often low-complexity models with little number of parameter, e.g., decision stump, linear model

Peter Flach: Machine Learning The Art and Science of Algorithms that Make Sense of Data
Slides CS6375: Machine Learning

Bias and Variance Decomposition Reasons

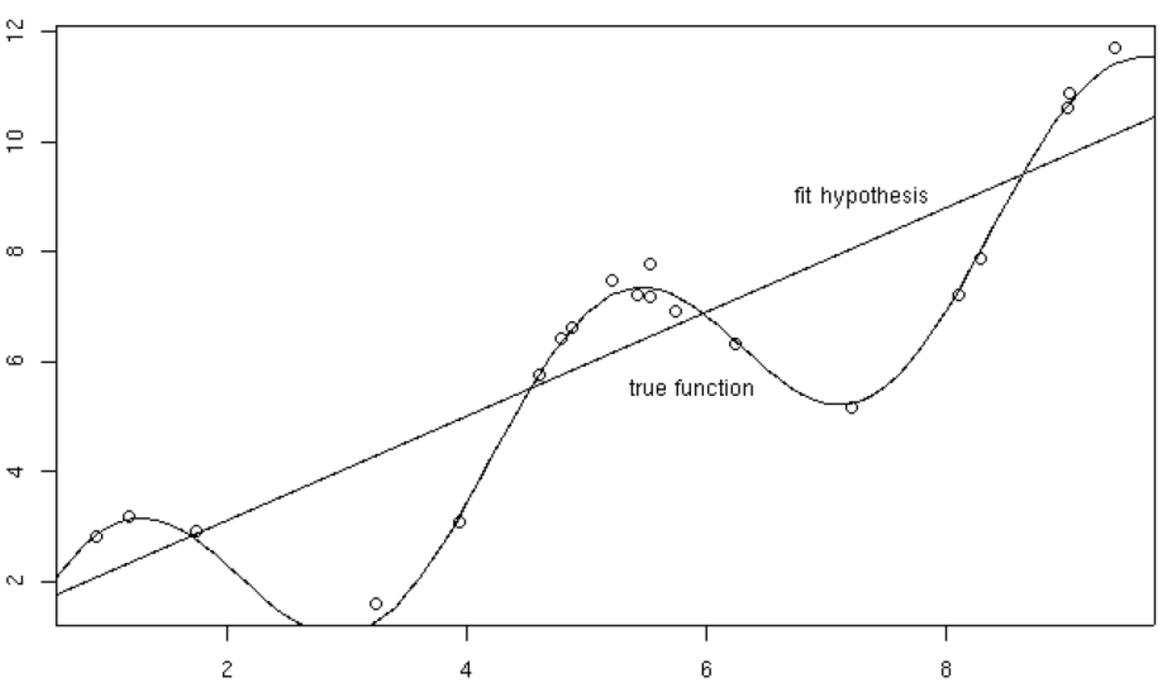
TECHNISCHE UNIVERSITÄT DARMSTADT

- What causes bias?
  - Inability to represent certain decision boundaries (linear hyperplanes, ..)
  - incorrect assumptions (independence assumption in naïve Bayes)
  - classifiers that are “too global”, “too general” (or too smooth)
  - e.g: single linear separator, small decision tree, a large k in k-NN
  - if the bias is high, the model is underfitting the data
- What causes variance?
  - making decision based on small subsets of the data
  - e.g., decision tree splits near the leaves
  - computational reasons, e.g., randomization in the learning algorithm
  - classifiers that are “too local” (or too nonlinear) can easily fit noisy data
  - learners that make sharp decisions can be unstable (change of decision boundary if one training example changes
  - if the variance is high, the model is likely overfitting the data

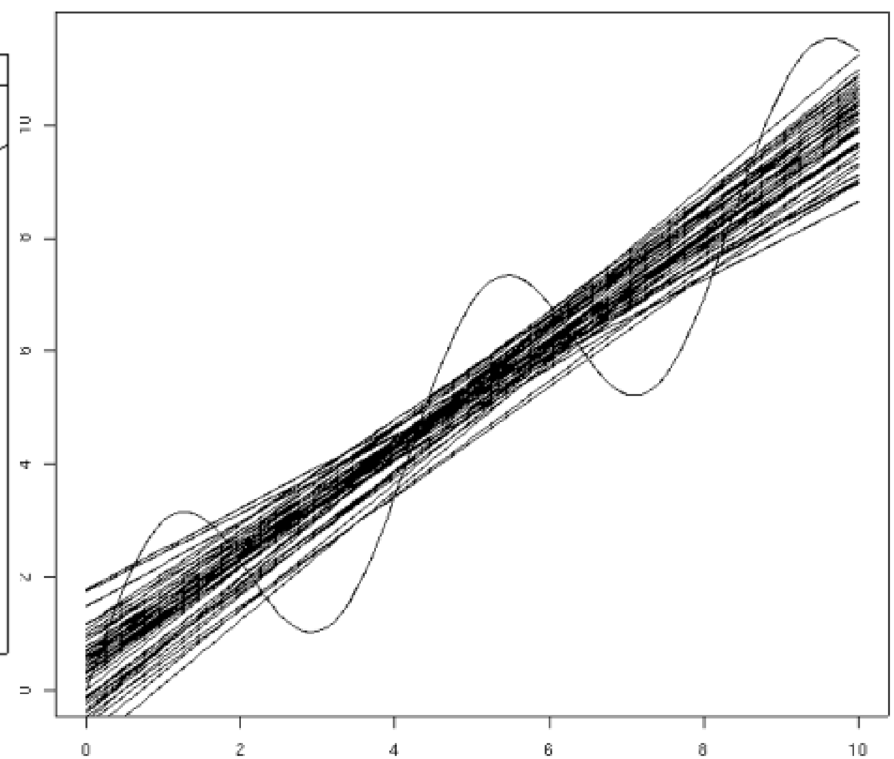
Peter Flach: Machine Learning The Art and Science of Algorithms that Make Sense of Data
Slides CS6375: Machine Learning

Bias and Variance Decomposition
Yet another view

Fitting noisy, linear-sinusal curve $f(x) = x + 2\sin(1.5x) + N(0,0.2)$ with linear function on 20 sampled training examples, repeated 50 times, gives:



50 fits (20 examples each)



following slides of Ruozzi: CS 6375

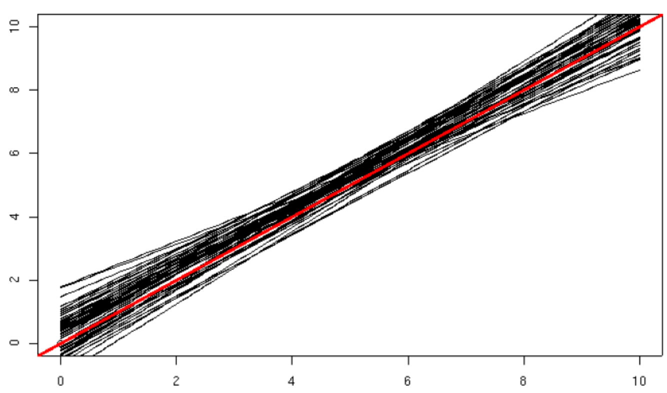
Bias and Variance Decomposition
Yet another view

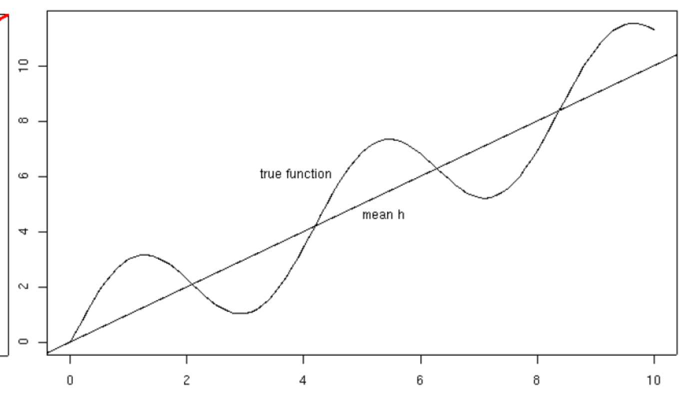
We can decompose error

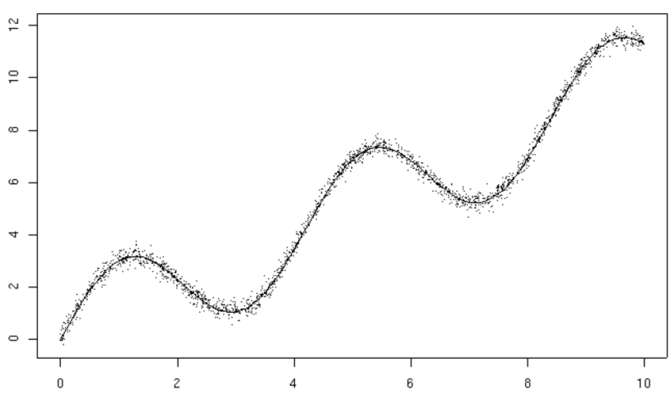
$$E\left[\left(y' - g_S(x')\right)^2\right]$$

into:

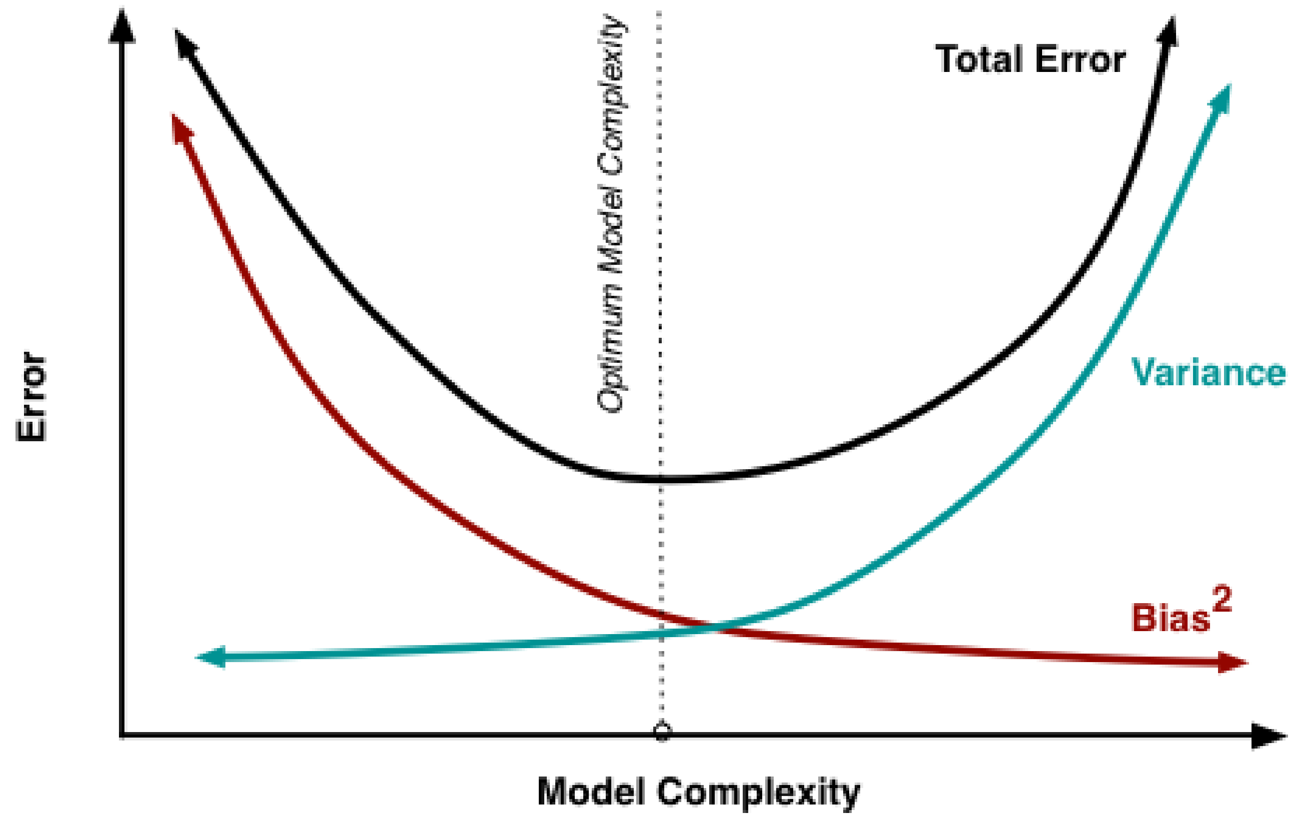
$$\underbrace{Var\left(g_S(x')\right)}_{\text{Variance}} + \underbrace{\left(E[g_S(x')] - f(x')\right)^2}_{\text{Bias}} + \underbrace{\sigma^2}_{\text{Noise}}$$







Bias and Variance Decomposition



Scott Fortman
Slides CS6375: Machine Learning

Ensemble Classifiers

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Idea**:
  - do not learn a *single* classifier but learn a *set of classifiers*
  - *combine the predictions* of multiple classifiers
- **Motivation**:
  - *reduce variance*: results are less dependent on peculiarities of a single training set
  - *reduce bias*: a combination of multiple classifiers may learn a more expressive concept class than a single classifier
- **Problem**:
  - Only one training set; where do multiple models come from?
- **Key step:**
  - formation of an ensemble of *diverse* classifiers from a single training set

Why do ensembles work?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Suppose there are 25 base classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$
  - Assume classifiers are independent
    - i.e., probability that a classifier makes a mistake does not depend on whether other classifiers made a mistake
    - **Note:** in practice they are not independent!
- Probability that the ensemble classifier makes a wrong prediction
  - The ensemble makes a wrong prediction if the majority of the classifiers makes a wrong prediction
  - The probability that 13 or more classifiers err is

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^{i} (1-\varepsilon)^{25-i} \approx 0.06 \ll \varepsilon$$

Why do ensembles work?

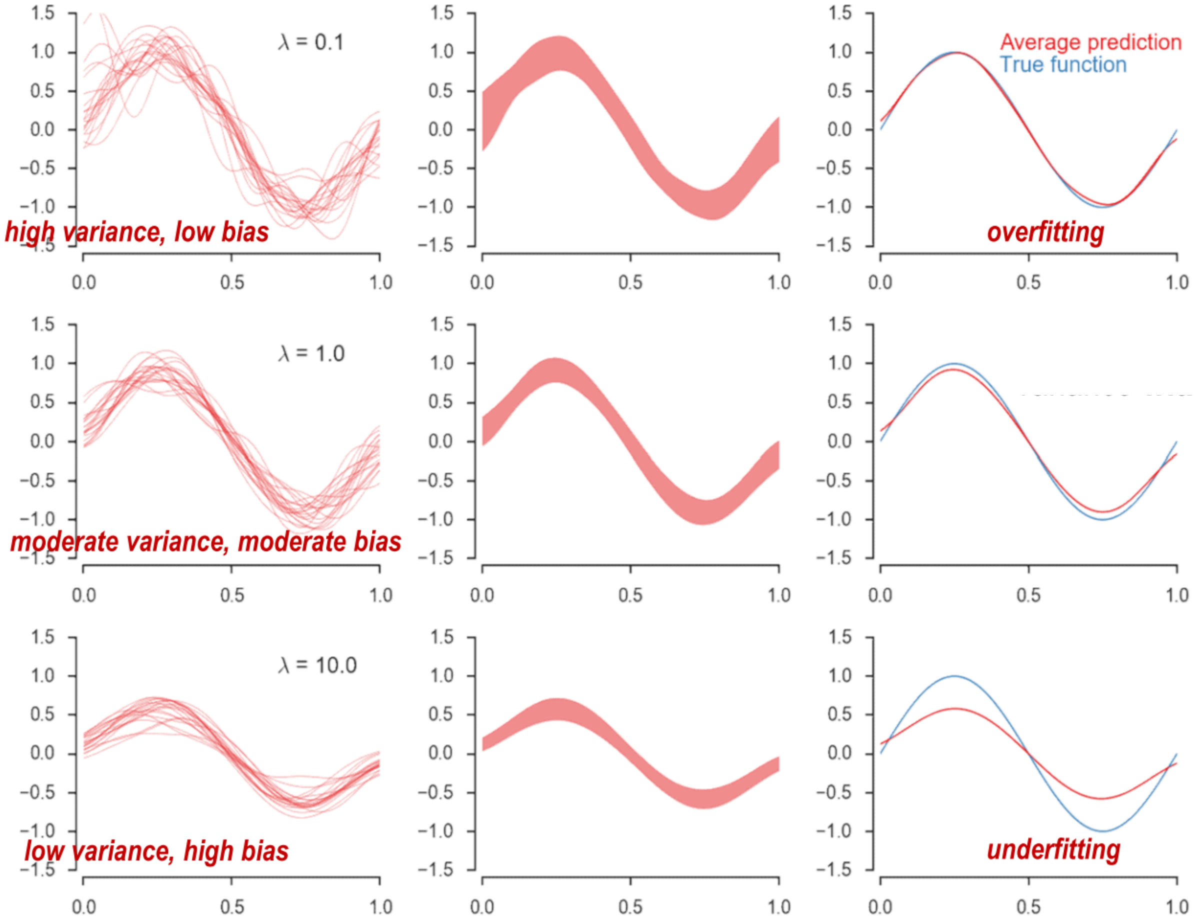
TECHNISCHE UNIVERSITÄT DARMSTADT

- When combining multiple independent and diverse decisions, random errors cancel each other out, correct decisions are reinforced
  - decision can come from weak learners: *at least* more accurate than random guessing
- Human ensembles are demonstrably better
  - How many jelly beans in the jar? individual estimates vs. group average
  - Who Wants to be a Millionaire: expert friend v. audience vote
  - “wisdom of the crowd”: crowd-sourcing

Can We Reduce Variance Without Increasing Bias?

TECHNISCHE UNIVERSITÄT DARMSTADT

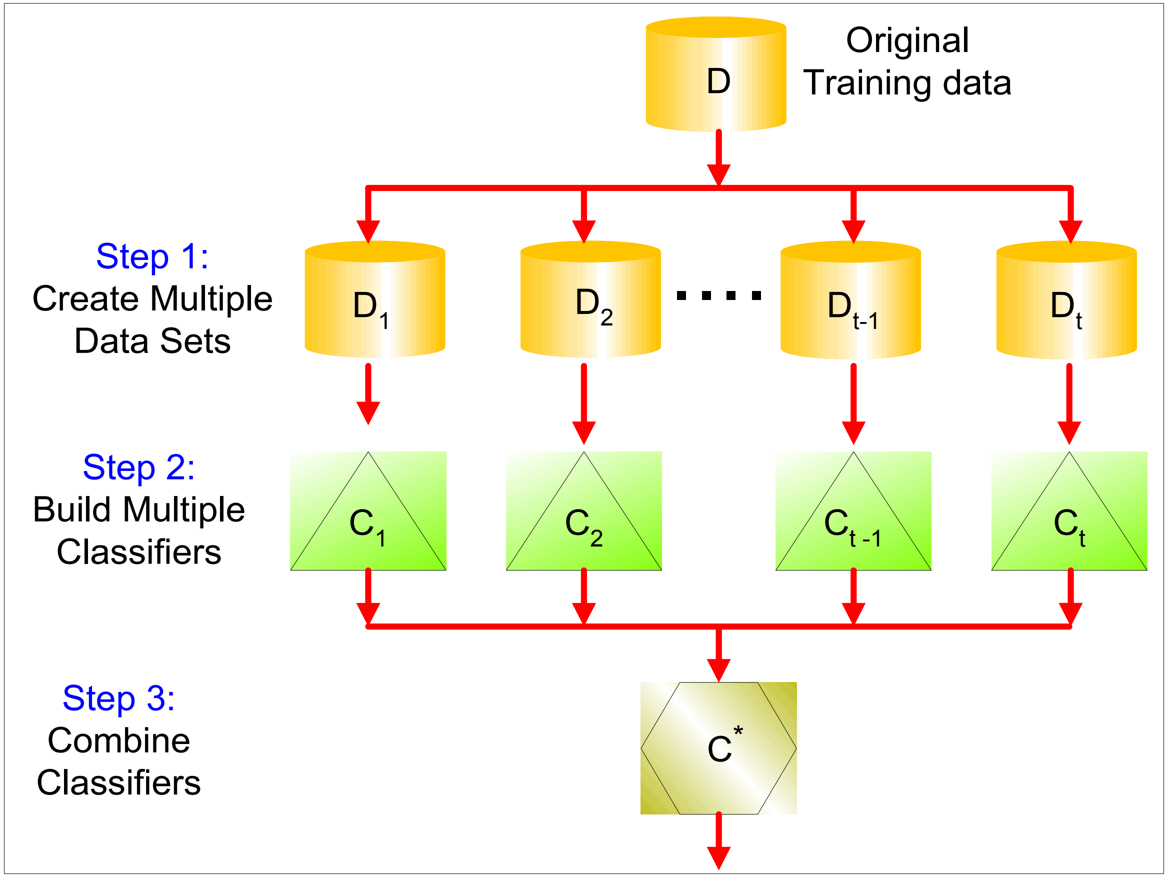
- model averaging can reduce variance without changing bias



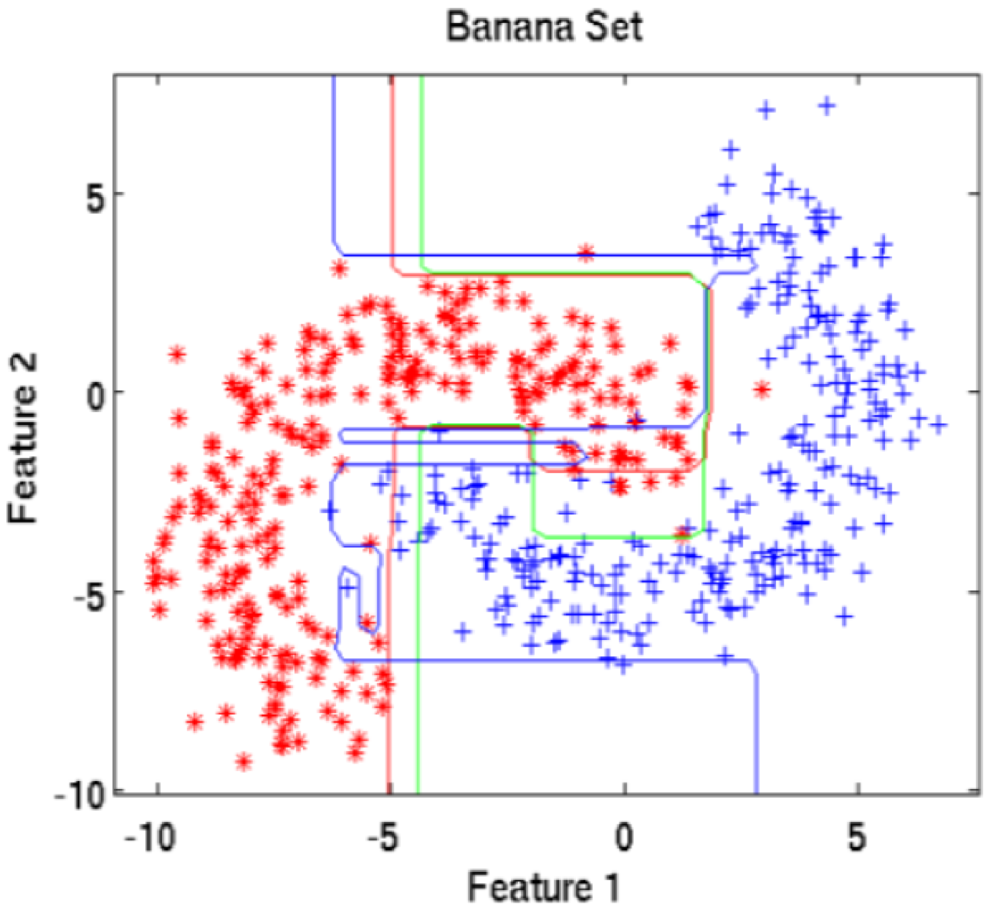
Slides CS6375: Machine Learning

Bagging Algorithm
Bootstrap AGGregatING

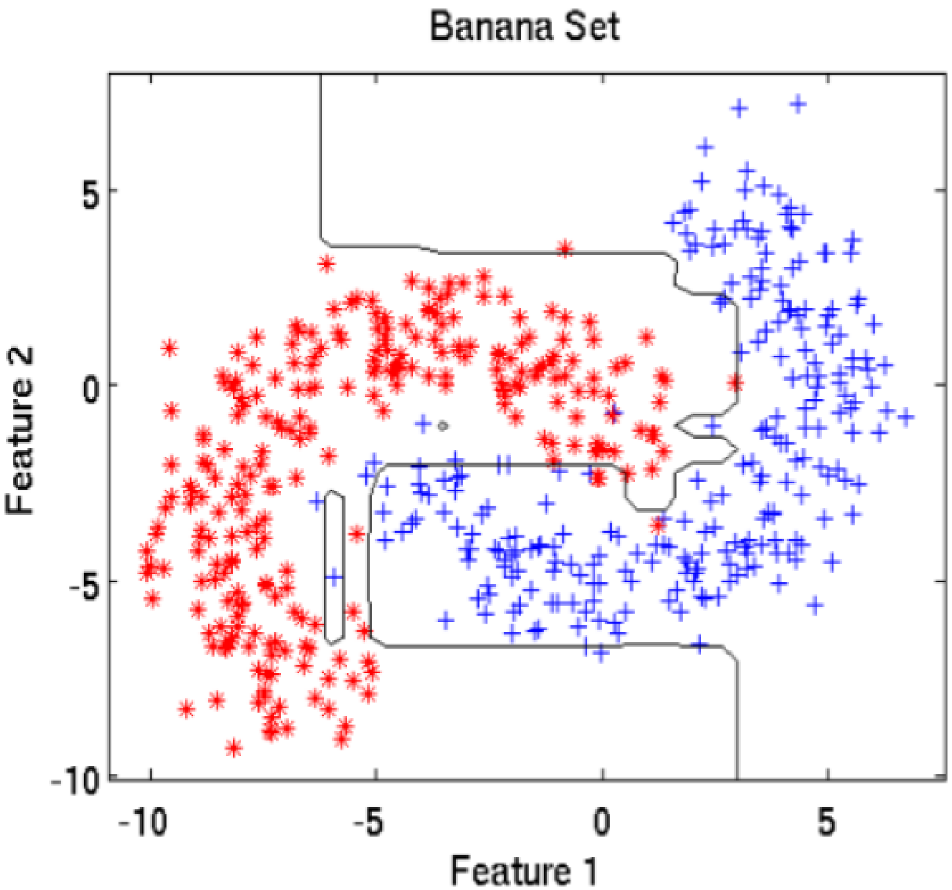
1. for $m = 1$ to t // t: #iterations
   a) draw (with replacement) a bootstrap sample D_m of the data D ($|D_m|=|D|$)
   b) learn a classifier C_m from D_m
2. for each test example
   a) apply all classifiers C_m
   b) predict the class that receives the highest number of votes



Bagging Algorithm
Example Models



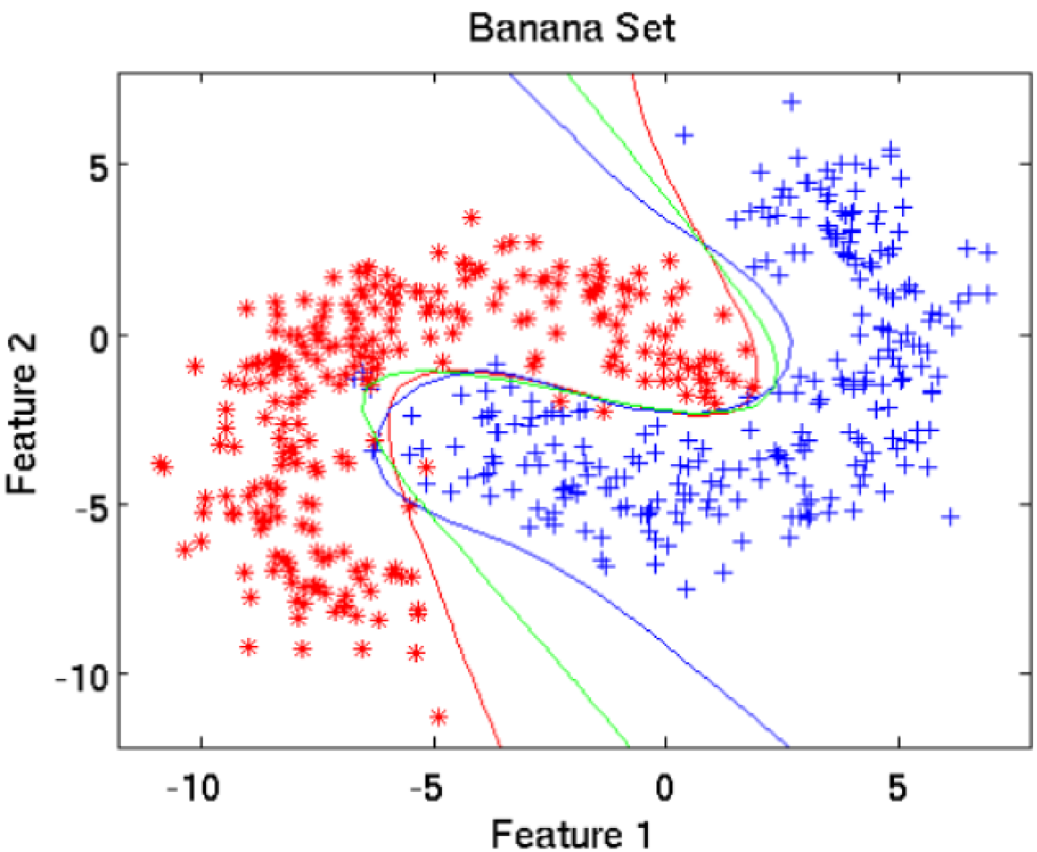
Three trees and final boundary overlaid



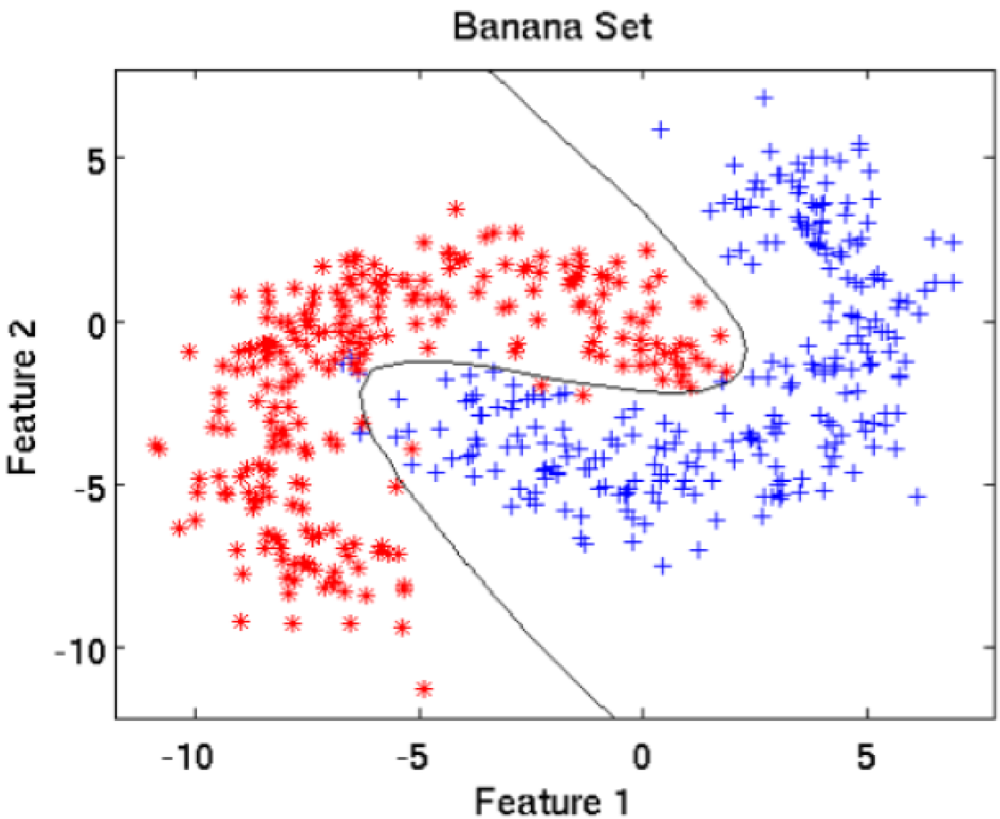
Final result from bagging all trees.

Amit Srinet, Dave Snyder: Bagging and Boosting

Bagging Algorithm
Example Models



Three neural nets generated with default settings [bpxnc]



Final output from bagging 10 neural nets

Bagging Algorithm Characteristics

How does bagging minimize the error?

- recall error term: $\mathbb{E}\left[\left(f(x)-\hat{f}(x)\right)^2\right] = \underbrace{\left(f(x)-\mathbb{E}\left[\hat{f}(x)\right]\right)^2}_{\text{bias: systematic error of all } \bar{f}(x)} + \underbrace{\mathbb{E}\left[\left(\hat{f}(x)-\mathbb{E}\left[\hat{f}(x)\right]\right)^2\right]}_{\text{variance: variability of } \bar{f}(x)}$
- due how it is built, $f^{\cdot}(x) = f^{\cdot}_{bag}(x)$ approximates the expectation of $\bar{f}(x)$
  → variance approximates 0
  (while leaving bias unchanged)
- in reality: bagging usually reduces variance and slightly increases bias

When to use bagging?

- for unstable base classifiers (small changes in data causes large changes in models)
- but could hurt stable classifiers

Variations

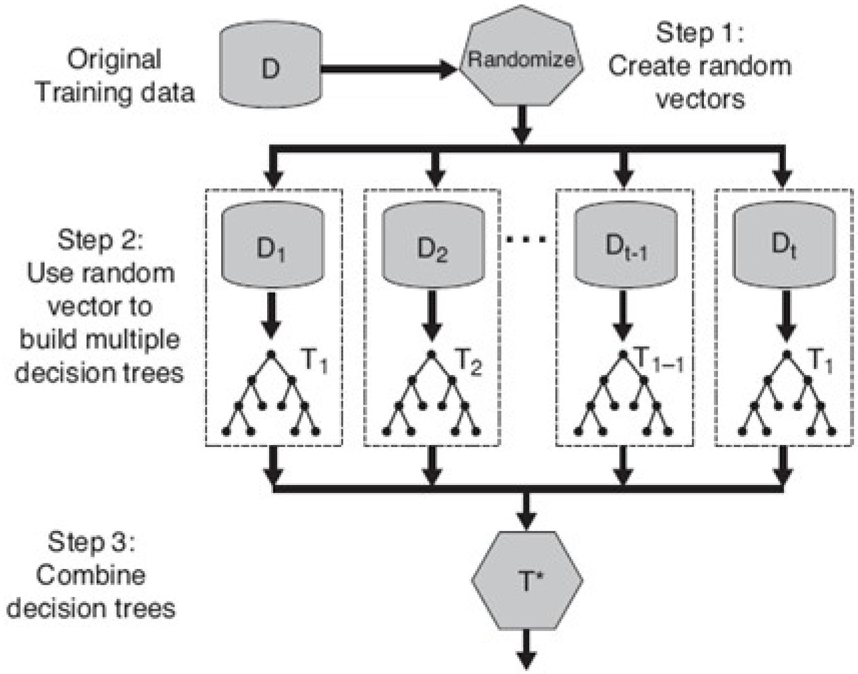
- size of subset, sampling w/o replacement, etc.
- sampling of features, learn a set of classifiers with different algorithms, etc.

Randomization

- Randomize the learning algorithm instead of the input data
- Some algorithms already have a random component
  - e.g.: initial weights in neural net
- Most algorithms can be randomized, e.g. greedy algorithms:
  - pick from the N best options at random instead of always picking the best options
  - e.g.: test selection in decision trees or rule learning
- Can be combined with bagging

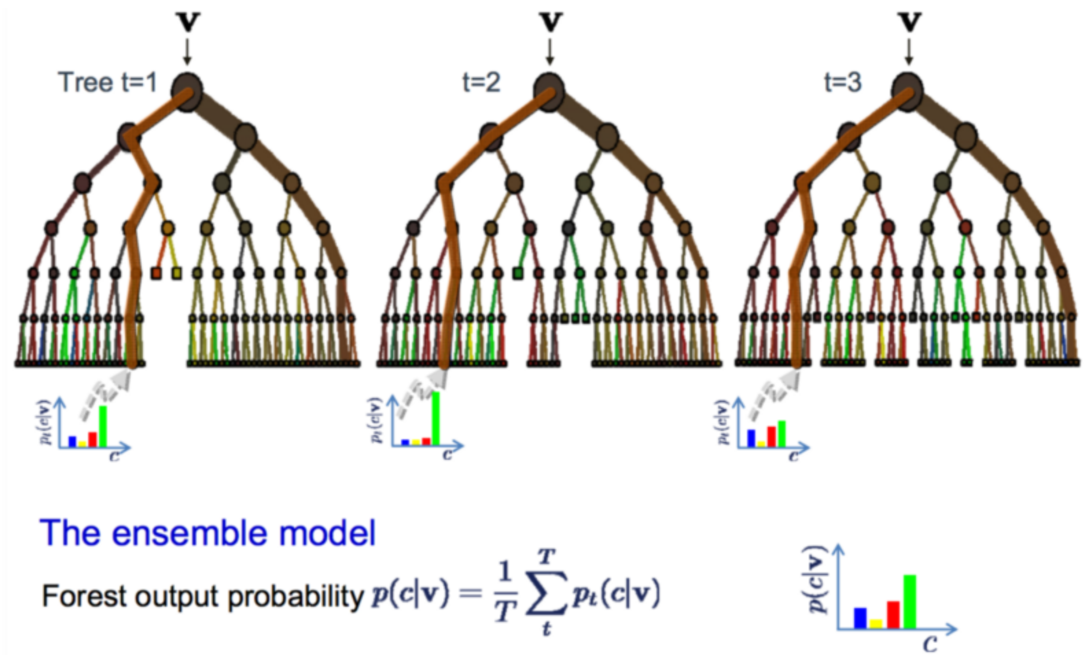
Random Forests

- Combines bagging and random attribute subset selection:
  - Build the tree from a bootstrap sample
  - Instead of choosing the best split among all attributes, select the best split among a random subset of k attributes
    - “random vector” in figure
    - is equal to bagging when k equals the number of attributes



Random Forests

- There is a bias/variance trade-off with k:
  - The smaller k, the greater the reduction of variance but also the higher the increase of bias
- Currently, one of the most successful methods in machine learning (see slides on decision trees)
- Interactive Demo: https://cs.stanford.edu/people/karpathy/svmjs/demo/demoforest.html



https://towardsdatascience.com/a-tour-of-the-top-10-algorithms-for-machine-learning-newbies-dde4edffae11

Boosting

- Basic Idea:
  - later classifiers focus on examples that were misclassified by earlier classifiers
  - weight the predictions of the classifiers with their error
- Realization
  - perform multiple iterations
    - each time using different example weights
  - weight update between iterations
    - *increase* the weight of *incorrectly* classified examples
    - this ensures that they will become more important in the next iterations
      (misclassification errors for these examples count more heavily)
  - combine results of all iterations
    - weighted by their respective error measures

Boosting – Algorithm AdaBoost.M1

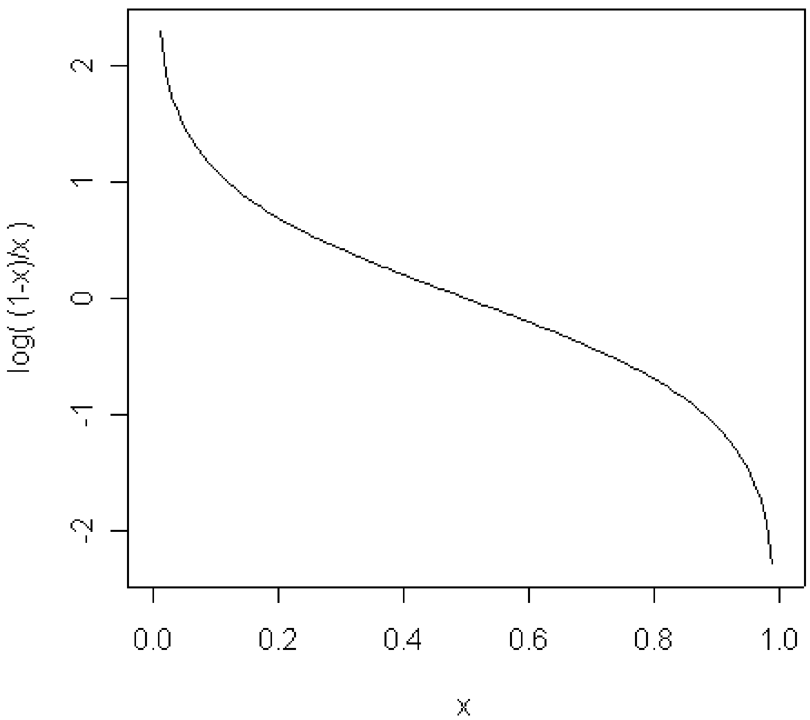
TECHNISCHE UNIVERSITÄT DARMSTADT

1. initialize example weights $w_i = 1/N \quad (i = 1..N)$
2. for $m = 1$ to t // t ... number of iterations
   a) learn a classifier C_m using the current example weights
   b) compute a weighted error estimate $$err_m = \frac{\sum w_i \text{ of all incorrectly classified } e_i}{\sum_{i=1}^{N} w_i}$$
   = 1 because weights are normalized
   c) compute a classifier weight $$\alpha_m = \frac{1}{2} \ln\left(\frac{1 - err_m}{err_m}\right)$$
   d) for all correctly classified examples e_i: $w_i \leftarrow w_i e^{-\alpha_m}$
   e) for all incorrectly classified examples e_i: $w_i \leftarrow w_i e^{\alpha_m}$
   f) normalize the weights w_i so that they sum to 1

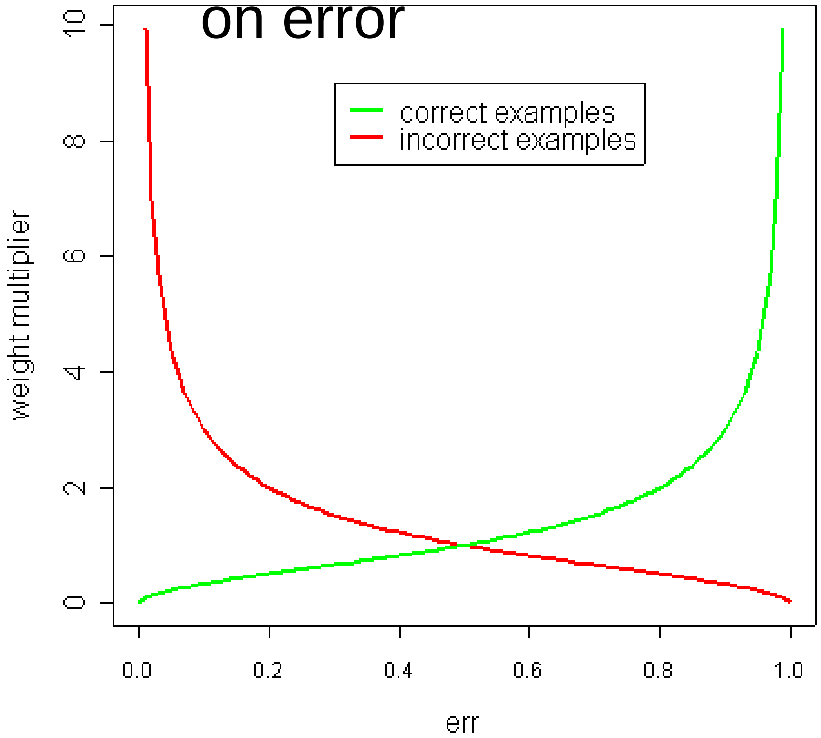
   update weights so that sum of correctly classified examples equals sum of incorrectly classified examples
3. for each test example
   a) try all classifiers C_m
   b) predict the class that receives the highest sum of weights α_m

Illustration of the Weights

- Classifier Weights α_m
  - differences near 0 or 1 are emphasized

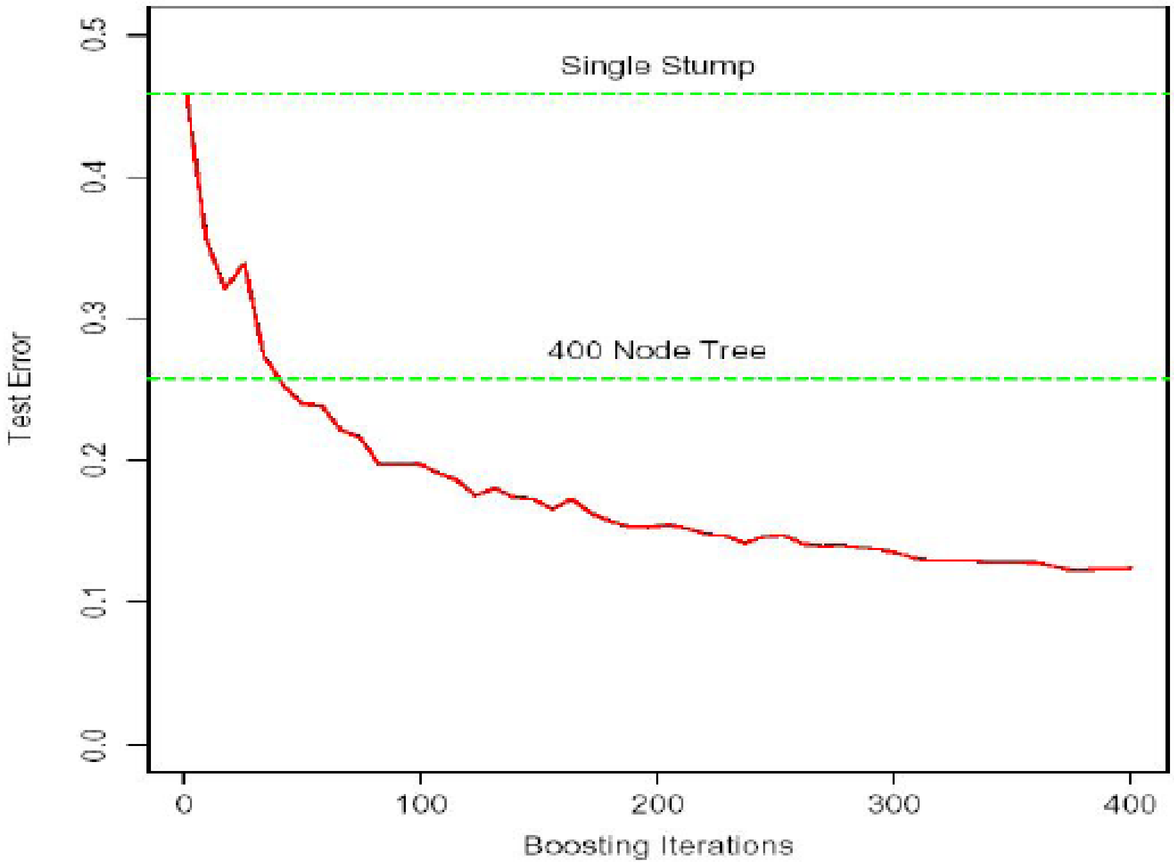


- Example Weights w_i
  - multiplier for correct and incorrect examples, depending on error



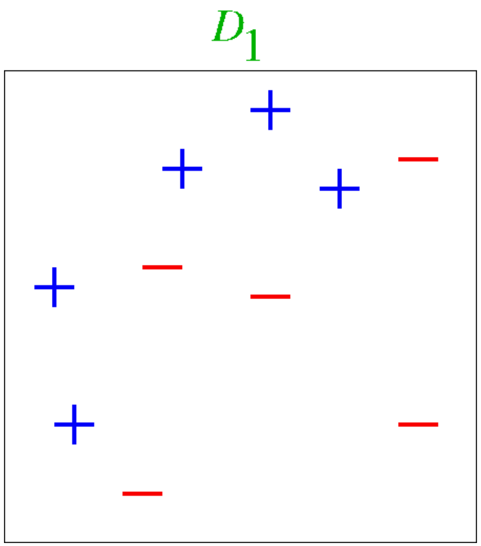
Boosting – Error rate example

- boosting of decision stumps on simulated data



from Hastie, Tibshirani, Friedman: The Elements of Statistical Learning, Springer Verlag 2001

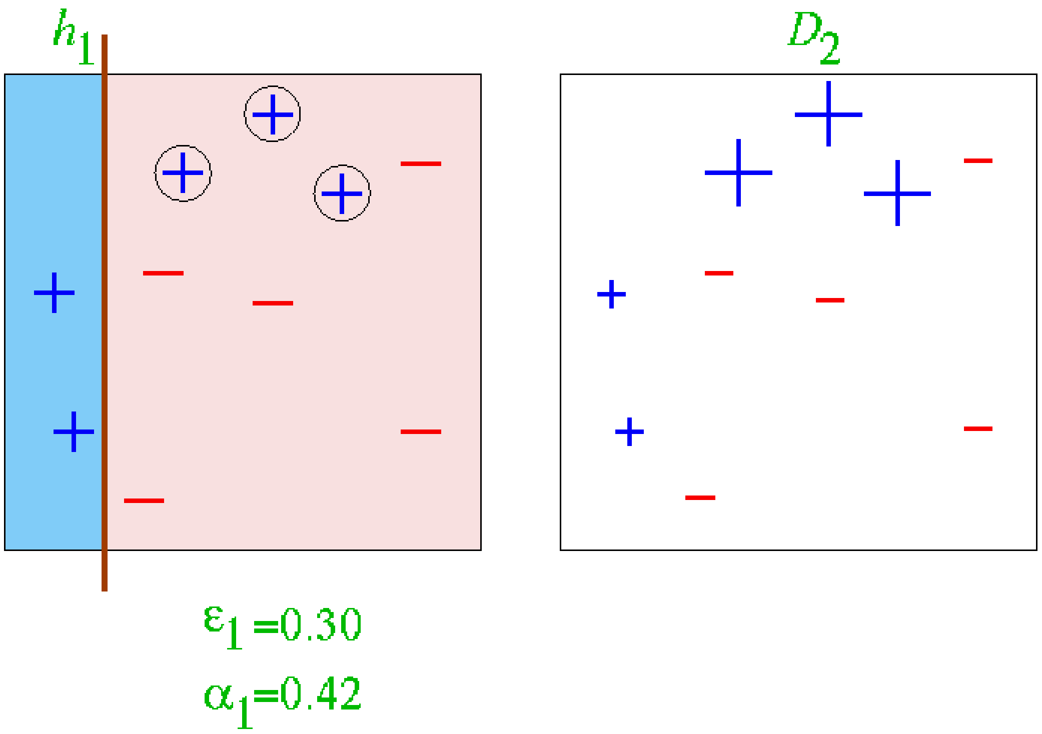
Toy Example



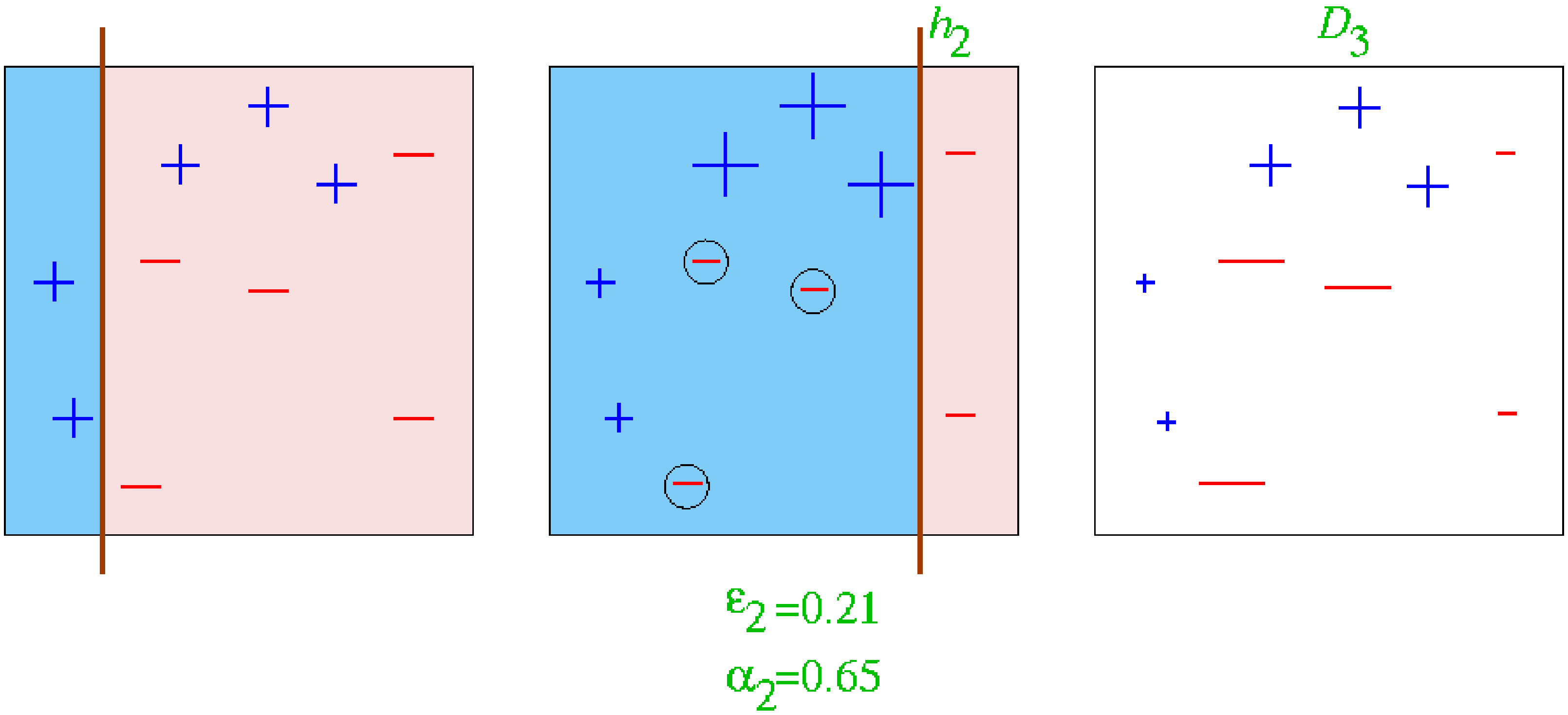
- An Applet demonstrating AdaBoost
  - http://www.cse.ucsd.edu/~yfreund/adaboost/

(taken from Verma & Thrun, Slides to CALD Course CMU 15-781, Machine Learning, Fall 2000)

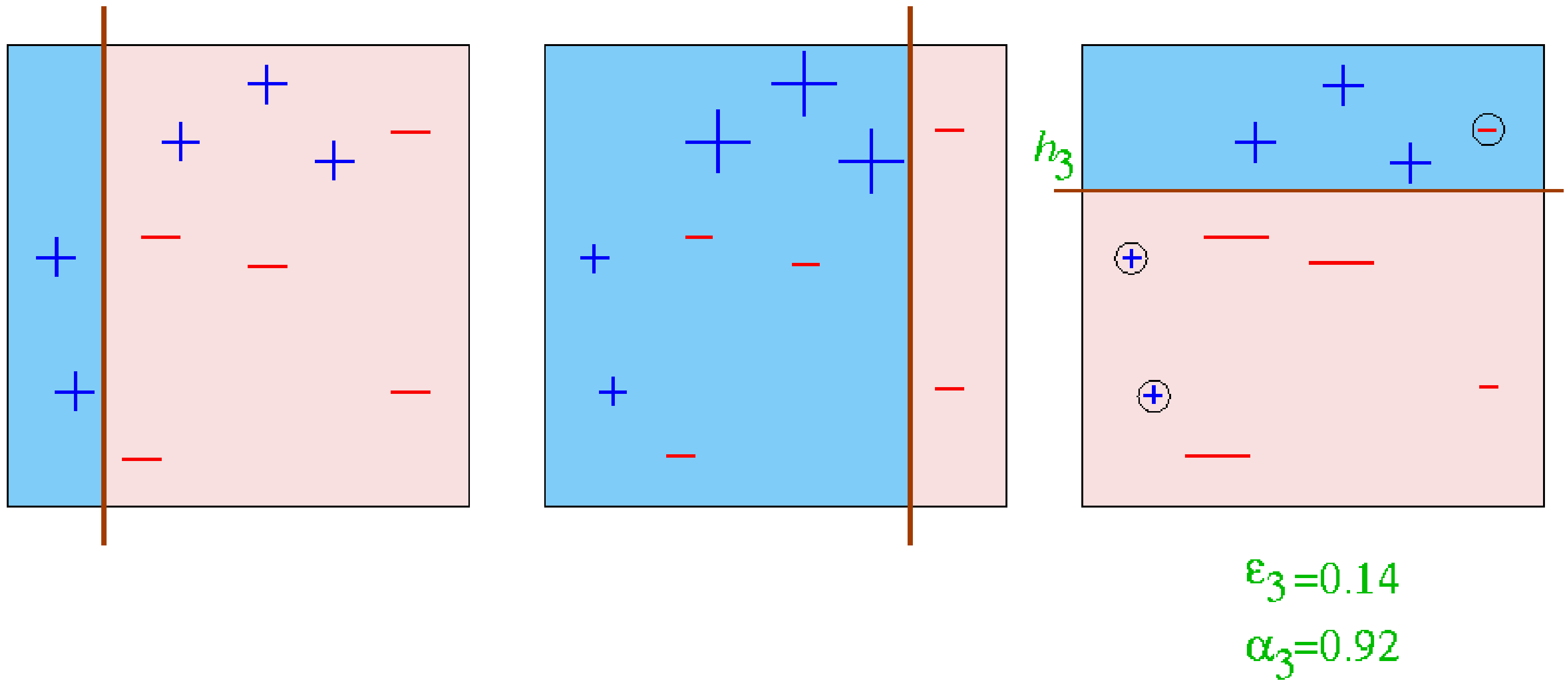
Round 1



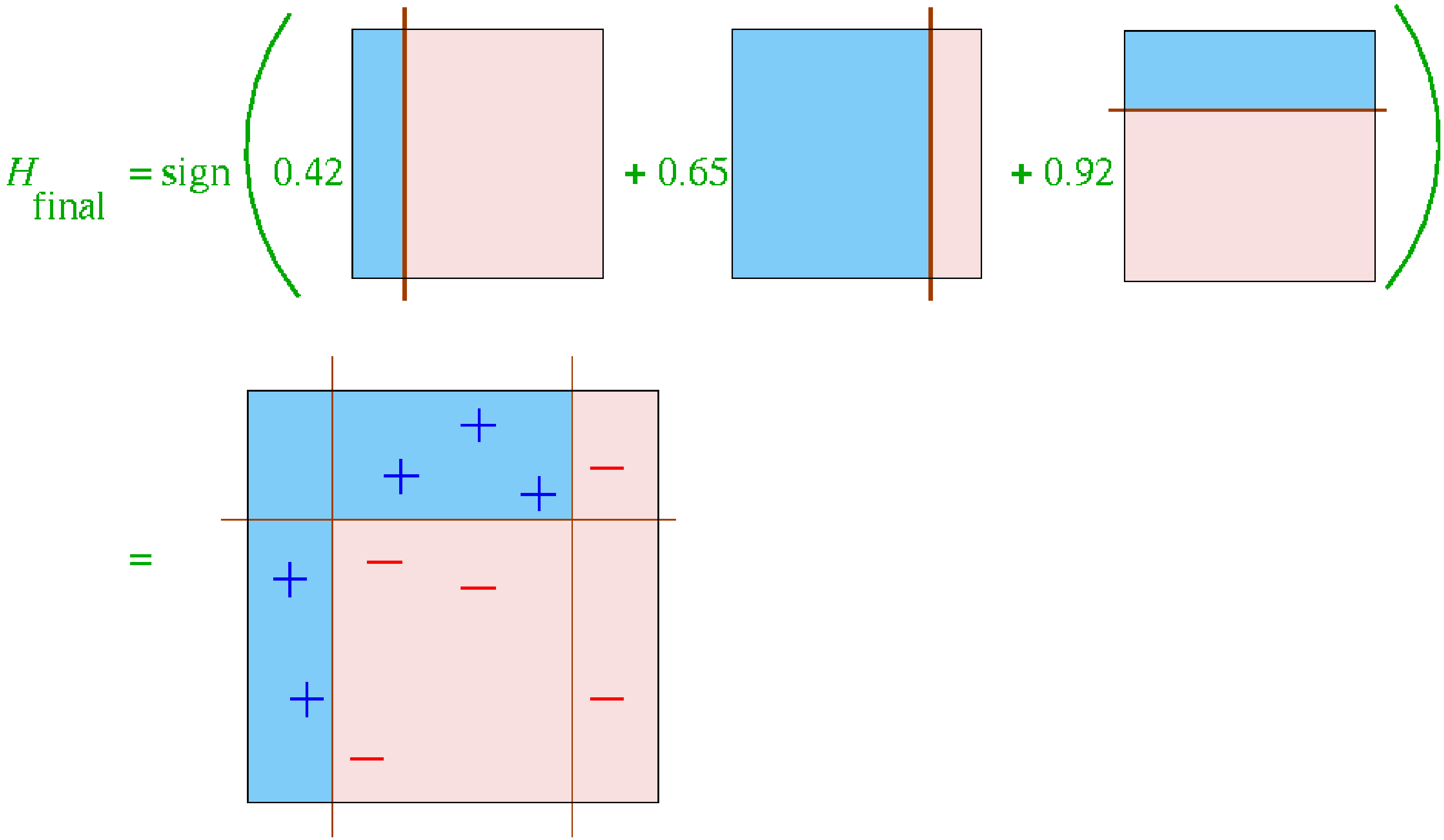
Round 2



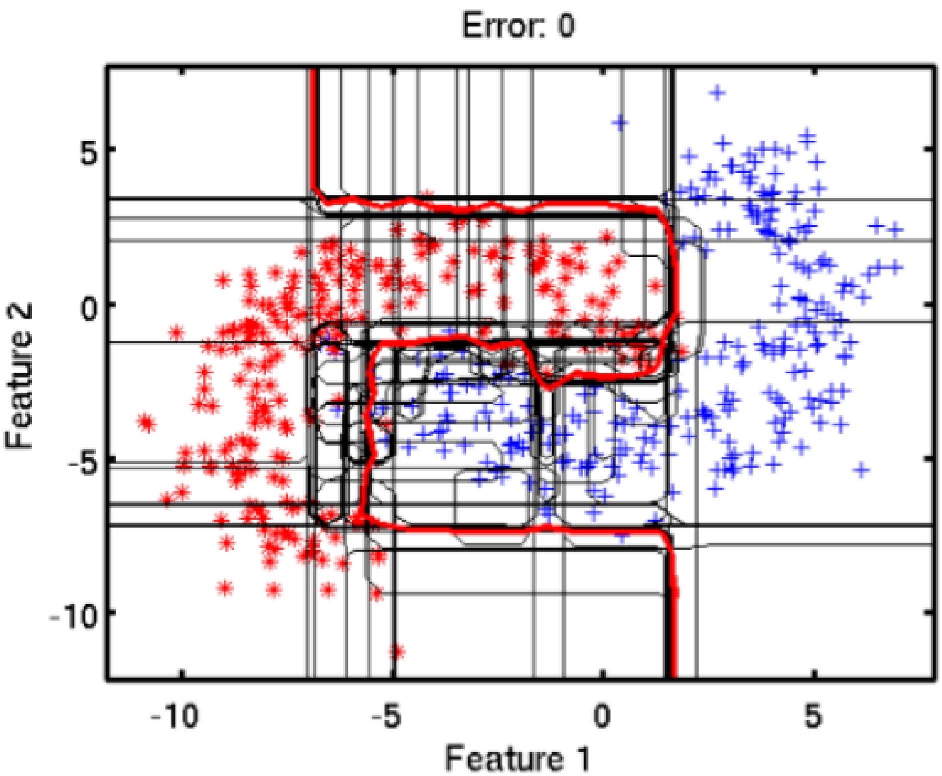
Round 3



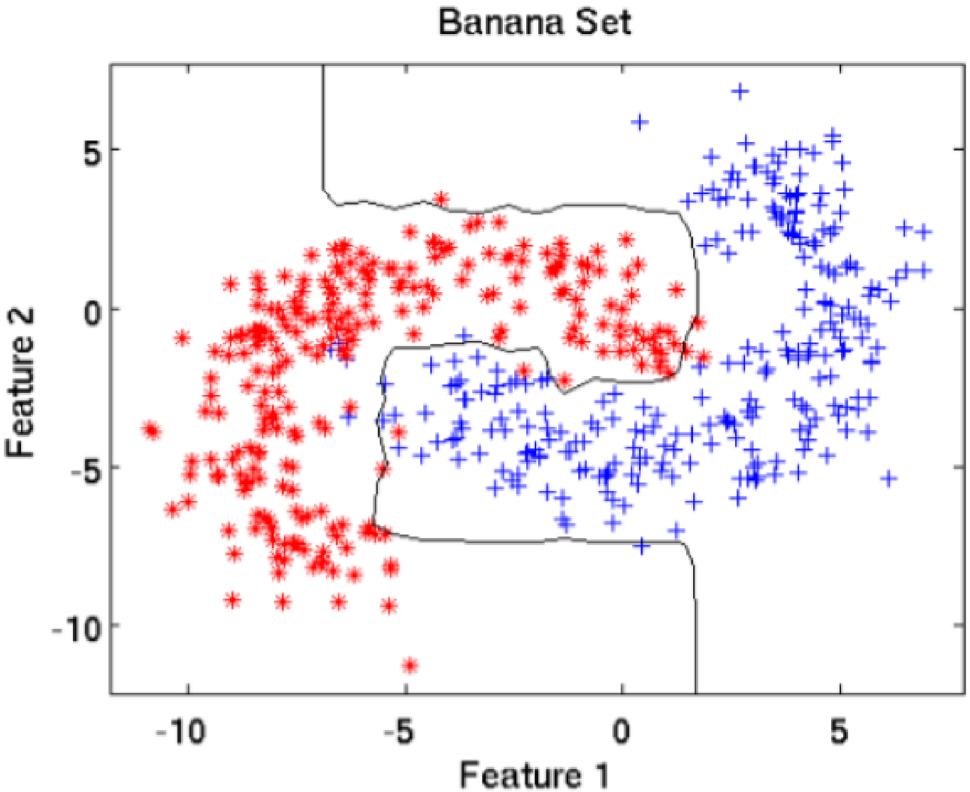
Final Hypothesis



Boosting
Example Models



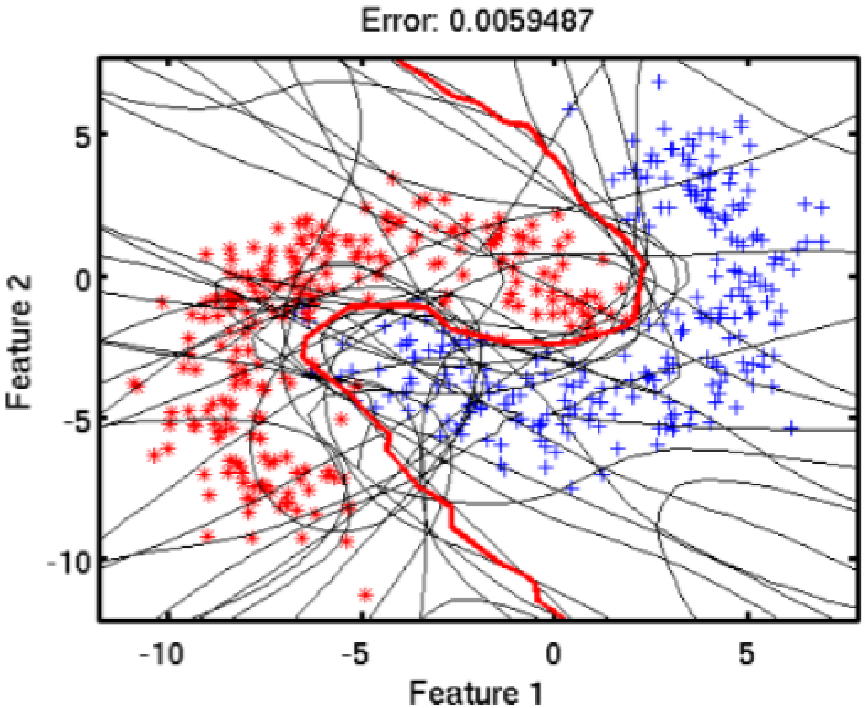
AdaBoost using 20 decision trees with default settings



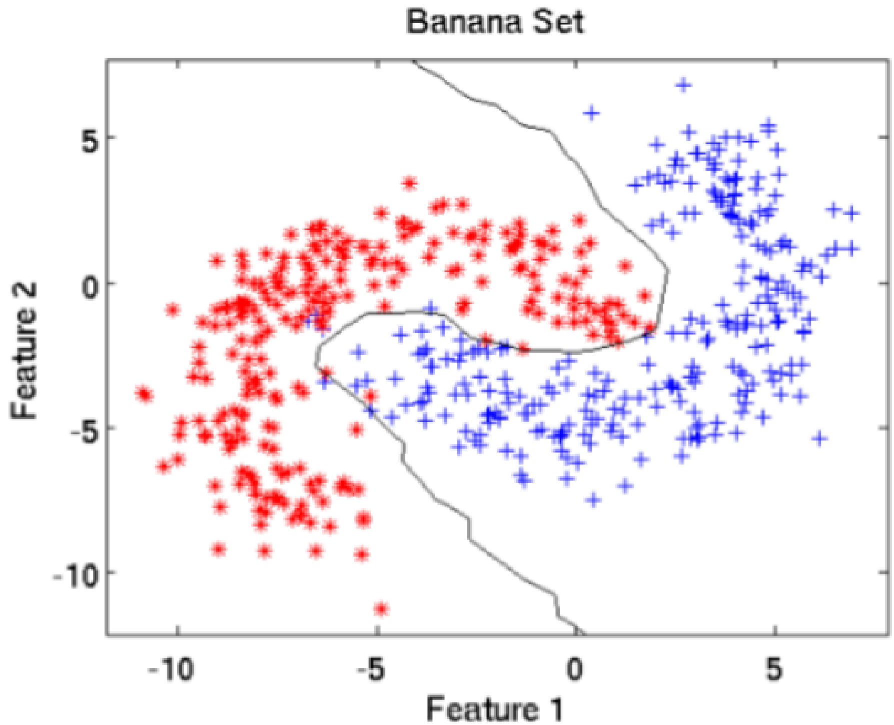
Final output of AdaBoost with 20 decision trees

Amit Srinet, Dave Snyder: Bagging and Boosting

Boosting
Example Models



AdaBoost using 20 neural nets [bpxnc] default settings



Final output of AdaBoost with 20 neural nets

Amit Srinet, Dave Snyder: Bagging and Boosting

Comparison Bagging/Boosting

- Bagging
  - noise-tolerant
  - produces better class probability estimates
  - not so accurate
  - statistical basis
  - each model may work by its own
- Boosting
  - very susceptible to noise in the data
  - produces rather bad class probability estimates
  - if it works, it works really well
  - based on learning theory (statistical interpretations are possible)
  - only first model is global, the subsequent ones are local and incremental

Additive regression

TECHNISCHE UNIVERSITÄT DARMSTADT

- It turns out that boosting is a greedy algorithm for fitting additive models
- More specifically, implements forward stagewise additive modeling
- Same kind of algorithm for numeric prediction:

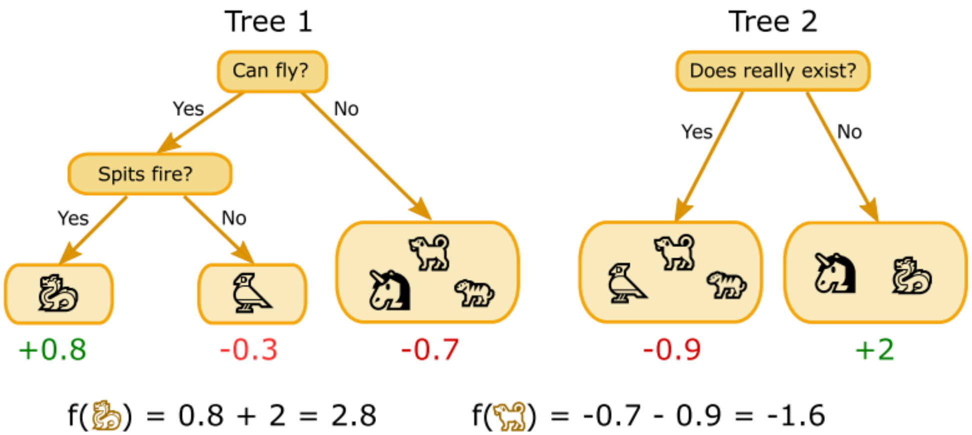
1. Build standard regression model (e.g. tree)
2. Gather residuals
3. learn model predicting residuals (e.g. tree)
4. goto 2.

- To predict, simply sum up individual predictions from all models

Additive regression
Gradient Boosting Trees

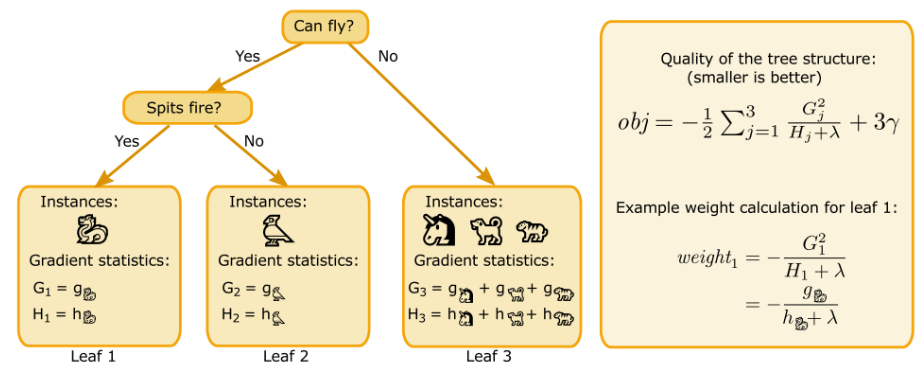
Reminder from slides on decision trees

- can use aleatory losses like MSE, logistic loss, ...
- splits determined by gain on gradient statistics
- regularization via tree size and model parameters in objective function



$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_c)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2$$



Simon Peter Bohlender, Dynamic Multilabel Classification using Gradient Boosted Trees

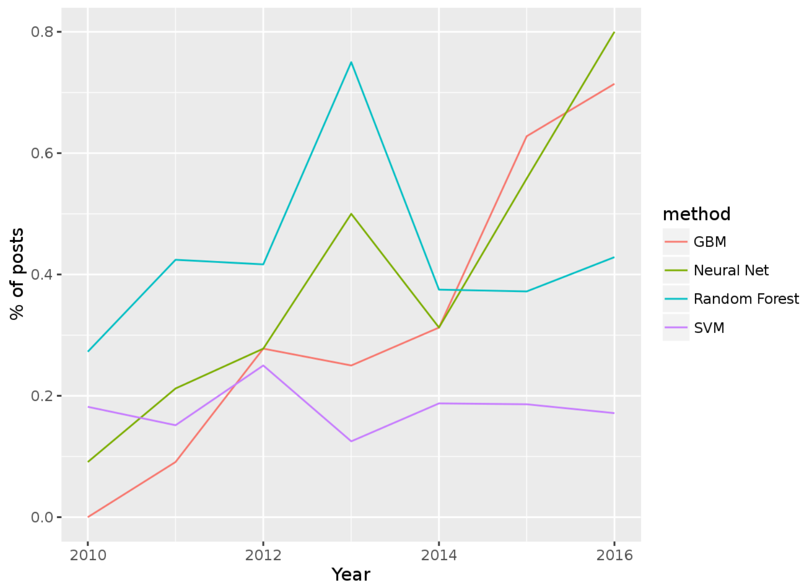
Additive regression
Gradient Boosting Trees

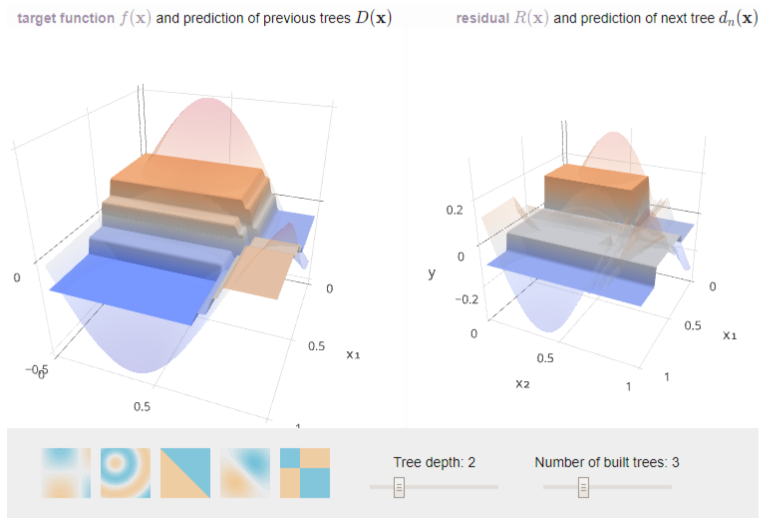
Reminder from slides on decision trees

- can use aleatory losses like MSE, logistic loss, ...
- splits determined by gain on gradient statistics
- regularization via tree size and model parameters in objective function

XGBoost

- one of the most successful machine learning algorithms in recent times
- won a lot of Kaggle competitions
- interactive visualizations: https://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html





https://xgboost.readthedocs.io/en/latest/tutorials/model.html
https://www.kaggle.com/msjgriffiths/r-what-algorithms-are-most-successful-on-kaggle/report

Combining Predictions

- voting
  - each ensemble member votes for one of the classes
  - predict the class with the highest number of vote (e.g., bagging)
- weighted voting
  - make a *weighted* sum of the votes of the ensemble members
  - weights typically depend
    - on the classifiers confidence in its prediction (e.g., the estimated probability of the predicted class)
    - on error estimates of the classifier (e.g., boosting)
- stacking
  - Why not use a classifier for making the final decision?
  - training material are the class labels of the training data and the (cross-validated) predictions of the ensemble members

Stacking

- Basic Idea:
  - learn a function that combines the predictions of the individual classifiers
- Algorithm:
  - train n different classifiers $C_1 \ldots C_n$ (the *base classifiers*)
  - obtain predictions of the classifiers for the training examples
  - form a new data set (the *meta data*)
    - **classes**
      - the same as the original dataset
    - **attributes**
      - one attribute for each base classifier
      - value is the prediction of this classifier on the example
  - train a separate classifier M (the *meta classifier*)

This is better done with cross-validation!

Stacking (2)

- Example:

Attributes			Class
x_{11}	$\ldots$	x_{1n_a}	t
x_{21}	$\ldots$	x_{2n_a}	f
$\ldots$	$\ldots$	$\ldots$	$\ldots$
x_{n_e1}	$\ldots$	$x_{n_en_a}$	t

training set

C_1	C_2	$\ldots$	C_{n_c}
t	t	$\ldots$	f
f	t	$\ldots$	t
$\ldots$	$\ldots$	$\ldots$	$\ldots$
f	f	$\ldots$	t

predictions of the classfiers

C_1	C_2	$\ldots$	C_{n_c}	Class
t	t	$\ldots$	f	t
f	t	$\ldots$	t	f
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
f	f	$\ldots$	t	t

training set for stacking

- Using a stacked classifier:
  - try each of the classifiers $C_1 \ldots C_n$
  - form a feature vector consisting of their predictions
  - submit these feature vectors to the meta classifier M

Summary: Forming an Ensemble

- Modifying the data
  - Subsampling
    - bagging
    - boosting
  - feature subsets
    - randomly feature samples
- Exploiting the algorithm characteristics
  - algorithms with random components
    - neural networks
  - randomizing algorithms
    - randomized decision trees
  - use multiple algorithms with different characteristics