Universität Stuttgart
KI – Institute for Artificial Intelligence

Analytic Computing

Machine Learning
9 Bagging and Boosting

Prof. Dr. Steffen Staab

Nadeen Fatallah
Daniel Frank
Akram Sadat Hosseini
Jiaxin Pan
Osama Mohamed
Arvindh Arunbabu
Tim Schneider
Yi Wang

https://www.ki.uni-stuttgart.de/

- Random forest based on slides by
  - Dr. Zeyd Boukhers, U. Koblenz-Landau

Learning Objectives

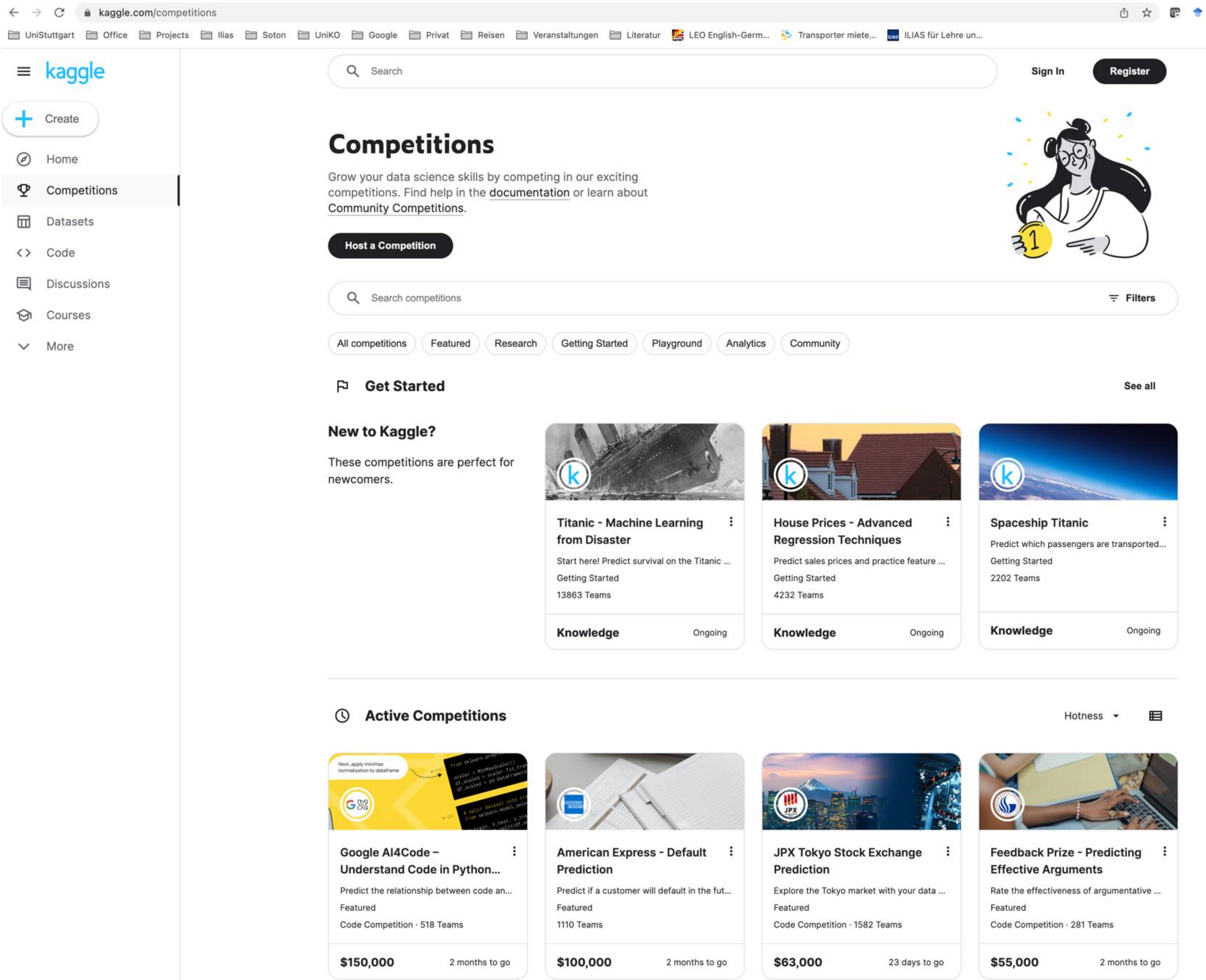
- Reconciling bias-variance trade-off by random forests
- Bagging – Boostrap sampling
- AdaBoost
  - Properties of AdaBoost
- Gradient Boosting
- Power of weak learners
  - learning from uncorrelated data / methods
- Strengths and weaknesses of different loss functions
  - including exponential loss

Meaning of boxes

explains the slide content

important take away

side note: nice to know


kaggle.com/competitions
UniStuttgart
Office
Projects
Ilias
Soton
UniKO
Google
Privat
Reisen
Veranstaltungen
Literatur
LEO English-Germ...
Transporter miete...
ILIAS für Lehre un...
kaggle
Search
Sign In
Register
Create
Home
Competitions
Datasets
Code
Discussions
Courses
More
Competitions
Grow your data science skills by competing in our exciting competitions. Find help in the documentation or learn about Community Competitions.
Host a Competition
Search competitions
Filters
All competitions
Featured
Research
Getting Started
Playground
Analytics
Community
Get Started
See all
New to Kaggle?
These competitions are perfect for newcomers.
Titanic - Machine Learning from Disaster
Start here! Predict survival on the Titanic ...
Getting Started
13863 Teams
Knowledge
Ongoing
House Prices - Advanced Regression Techniques
Predict sales prices and practice feature ...
Getting Started
4232 Teams
Knowledge
Ongoing
Spaceship Titanic
Predict which passengers are transported...
Getting Started
2202 Teams
Knowledge
Ongoing
Active Competitions
Hotness
Google AI4Code – Understand Code in Python...
Predict the relationship between code an...
Featured
Code Competition · 518 Teams
$150,000
2 months to go
American Express - Default Prediction
Predict if a customer will default in the fut...
Featured
1110 Teams
$100,000
2 months to go
JPX Tokyo Stock Exchange Prediction
Explore the Tokyo market with your data ...
Featured
Code Competition · 1582 Teams
$63,000
23 days to go
Feedback Prize - Predicting Effective Arguments
Rate the effectiveness of argumentative ...
Featured
Code Competition · 281 Teams
$55,000
2 months to go

Which machine learning methods win most Kaggle competitions?

- Not deep learning, but
- ensemble methods
  - previously: Random Forest
  - nowadays: XGBoost
  - maybe in the future: ensemble of neural networks
- Currently: XGBoost as the method to look to for tabular data
  - we do not consider XGBoost in this method,
    but it is a modification of Adaboost, which we consider

Reasons

- Table: (mostly) heterogeneous data
- Text / image: homogeneous data

- Deep learning: linear combinations (plus activation function) over homogeneous data
- Tree algorithms: individual treatments of table columns

1 Bagging and Boosting

Prediction Error

At an input x_0 using squared error loss and a regression fit $\hat{f}$ the prediction error is:

$$\mathrm{Err}(x_0) = E\left[\left(y_0 - \hat{f}(x_0)\right)^2\right] =$$

$$= E\left[\left(Y - \hat{f}(x_0)\right)^2 | X = x_0\right] =$$

$$= \sigma_\varepsilon^2 + \left[E\hat{f}(x_0) - f(x_0)\right]^2 + E\left[\hat{f}(x_0) - E\hat{f}(x_0)\right]^2 =$$

$$= \sigma_\varepsilon^2 + \mathrm{Bias}^2\left(\hat{f}(x_0)\right) + \mathrm{Var}\left(\hat{f}(x_0)\right) =$$

$$= \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}$$

Also cf. bias-variance trade-off in 4 Linear Regression

What if variance of a prediction function is too high?

- What if variance of a prediction function is too high?

For instance, decision trees tend to have small bias, but large variance

$\Longrightarrow$ bagging

1. Choose data samples
2. Learn one predictor on this data sample
3. **Regression**: average the results of the predictors
   Classification: vote for classes by all the predictors

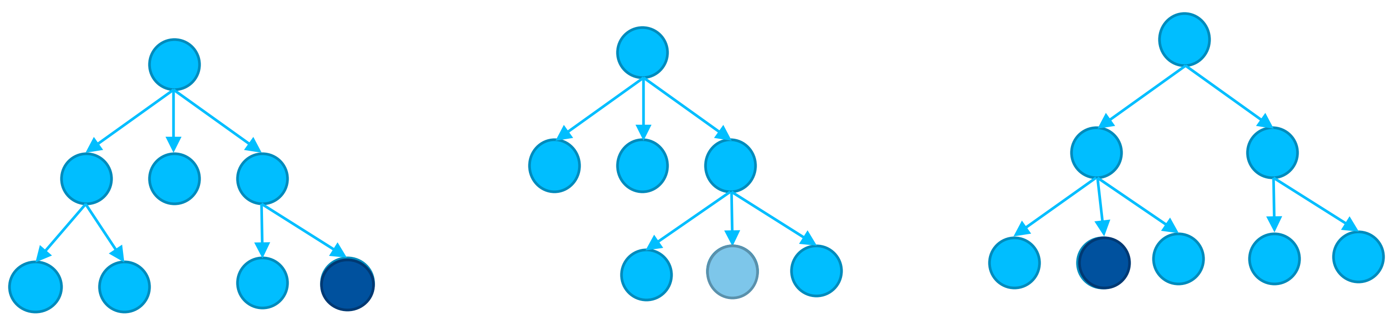
$\Longrightarrow$ boosting

1. Iterate
   (i) weight training data, (ii) choose weak predictor
2. **Regression/Classification**: weighted average of predictors

2 Constructing a Random Forest

Random forest

- It constructs multiple *decision trees*
- The final decision is made based on the majority votes of all trees.



Reducing variance

- Averaging B independent identically distributed (i.i.d.) random variables each with mean μ and variance σ^2 results in
  - mean μ (this means: unchanged bias)
  - variance $\frac{1}{B}\sigma^2$ (this means: lower variance)

- If random variables are pairwise correlated with ρ, new variance will be

$$\left(\rho + \frac{1-\rho}{B}\right)\sigma^2$$

Bagging (also *bootstrap aggregation*)

- What if variance of a prediction function is too high?

$\Longrightarrow$ bagging

1. Choose data samples
2. Learn a predictor on every data sample
3. Regression: average the results of the predictors
   Classification: vote for classes by all the predictors

$\Longrightarrow$ random forests as a special method for bagging uncorrelated trees

Random forest

- Define the number of trees B
- For $b = 1$ to B
  - Draw a bootstrap sample Z^* of size N^* from the original data set with size N.
    - *Bootstrap: random sampling with replacement.
  - Grow a *random-forest tree* T_b using the Z^* sample.

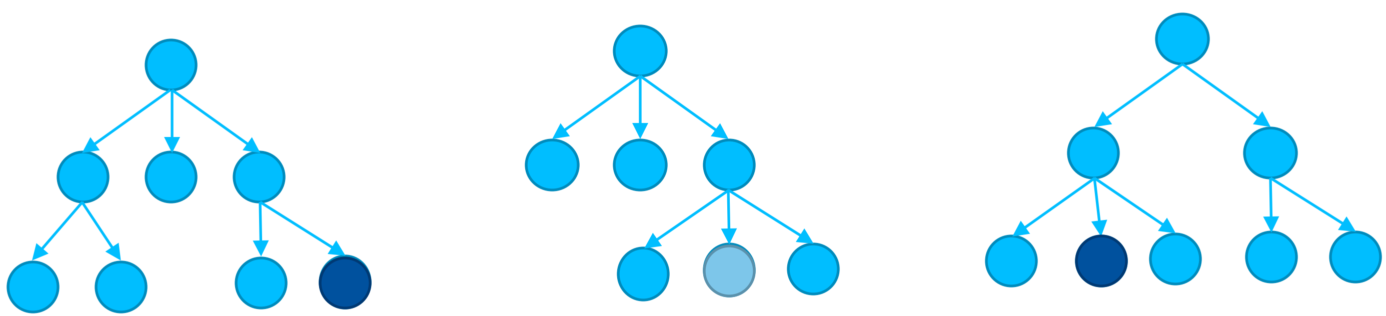
    For each node, repeat until minimal node size n_{min} is reached:
    - Select l^* attributes at random from all attributes.
    - Using *Gain Ratio*, split the node into children nodes.
- Output the ensemble of trees $\{T_b\}_{b=1}^{B}$

$l^* < \sqrt{l}, \text{even } 1$ uncorrelates the trees

Using random forest for classification

- Let $\hat{f}_b(\boldsymbol{x})$ be the class prediction of $\boldsymbol{x}$ by the b tree,

$$\hat{f}(\boldsymbol{x}) = \underset{i}{\operatorname{argmax}} \sum_{b=1}^{B} [\hat{f}_b(\boldsymbol{x}) = i]$$



Generalization Error of an ensemble of Decision Trees

- Notes:
  - The bias of the ensemble is identical to the bias of a randomized tree but higher than the bias of a non-randomized tree.
  - Stronger randomization: $\mathrm{var}(X) \to 0$
  - Weaker randomization: $\mathrm{var}(X) \to$ the variance of a non-randomized tree.

✓Randomization increases bias but decreases the variance of the ensemble of trees.

✓Find the right bias-variance trade-off.

Advantages

- Better accuracy than a decision tree.
- Robustness to outliers and missing data.
- Robustness to irrelevant attributes.
- Non-parametric (completely random)
- Invariant to feature scaling and types.
- Reduces overfitting.
- High accuracy, especially for large data sets.

- Interpretability ???

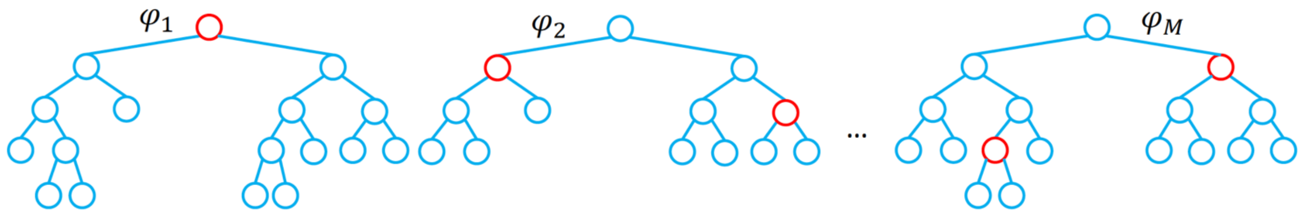
3 Interpreting Random Forests

Attribute importance score: Mean Decrease of Accuracy

- Consider the out-of-bag samples (which were not sampled in the bootstrapped set).
  - Of course, we may use a separate set.
- Consider the corresponding trees.
- Permute the value of the attribute (to be assessed) with random noise.
- Consider an evaluation metric (e.g. accuracy)
- Compute the mean decrease of accuracy over all corresponding trees.

➢ The attribute is important when the MDA is high

Attribute importance score: Mean Decrease of Impurity



Importance of variable X_j for an ensemble of B trees φ_m is :

$$\mathrm{Imp}(X_j) = \frac{1}{B} \sum_{m=1}^{B} \sum_{t \in \varphi_m} 1(j_t = j) \Big[p(t) \Delta i(t) \Big],$$

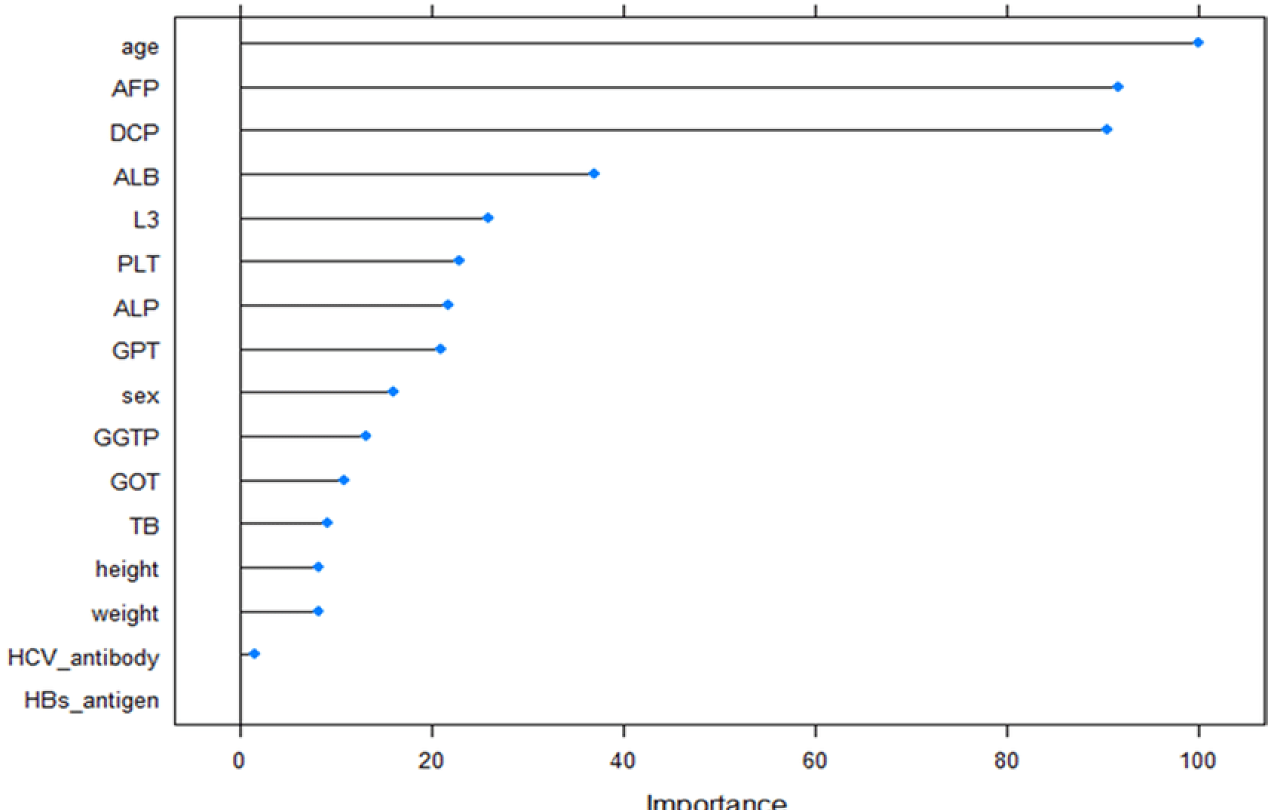
where j_t denotes the variable used at node t, $p(t) = N_t / N$ and $\Delta i(t)$ is the impurity reduction at node t :

$$\Delta i(t) = i(t) - \frac{N_{t_L}}{N_t} i(t_L) - \frac{N_{t_r}}{N_t} i(t_R)$$

Attribute importance scores (2)

- Mean Decrease in Impurity (MDI):
  - The importance of an attribute x_j is measured as:
    - $Imp(x_j) = \frac{1}{B}\sum_{b=1}^{B}\sum_{t:b(t)=x_j} P(t) * \Delta I(t)$,
    - Where $P(t) = \frac{N_t}{N}$
  - The intuition is that an attribute is important when:
    - It decreases many impurities
    - It is used to split nodes with many samples.
    - It is used many times.
- Compared to MDA, MDI is widely used because:
  - It is faster and easier to compute.
  - Experiences showed that it correlates well with MDA.

Example of MDI



Source: Sato, M., Morimoto, K., Kajihara, S., Tateishi, R., Shiina, S., Koike, K., & Yatomi, Y. (2019). Machine-learning Approach for the Development of a Novel predictive Model for the Diagnosis of Hepatocellular Carcinoma. *Scientific reports*, *9*.

Computational complexity

	Training	Prediction
Decision Tree	$O(l * N \log^2(N))$	$O(l)$
Random Forest	$O(B * \hat{l} * \hat{N} \log^2(\hat{N}))$	$O(B * \hat{l})$
Extra Tree	$O(B * \hat{l} * N \log(N))$	$O(B * \hat{l})$

- $\hat{l}$: the number of variables randomly drawn at each node.
- B: the number of trees
- $\hat{N} = 0.632N$

Take away

On random forests

- Sampling the data is not good enough, sampling of attributes is also needed

In general

- When approaching a new machine learning problem:
  - try simple methods first
  - random forest is a very good simple method to start with

4 Explainable Machine Learning

Local feature attributions by Shapley values

Coalition game theory: allocate the surplus generated by grand coalition to players

The Shapley value $\mathcal{S}_j$ for the j 'th player is defined via a

$$\mathcal{S}_j(\text{val}) = \sum_{T \subseteq N \setminus \{j\}} \frac{|T|!\,(p - |T| - 1)!}{p!} (\text{val}(T \cup \{j\}) - \text{val}(T))$$

- $N = \{1, \dots, p\}$ is the set of features; x is the vector of the instance to be explained,
- $\text{val}_{f,x}(T)$ represents the prediction for the feature values in T that are marginalized over features that are not included in T :

$$\text{val}: 2^N \to \mathbb{R}, \text{val}_{f,x}(T) = E_{X|X_T = x_T}[f(X)] - E_X[f(X)]$$

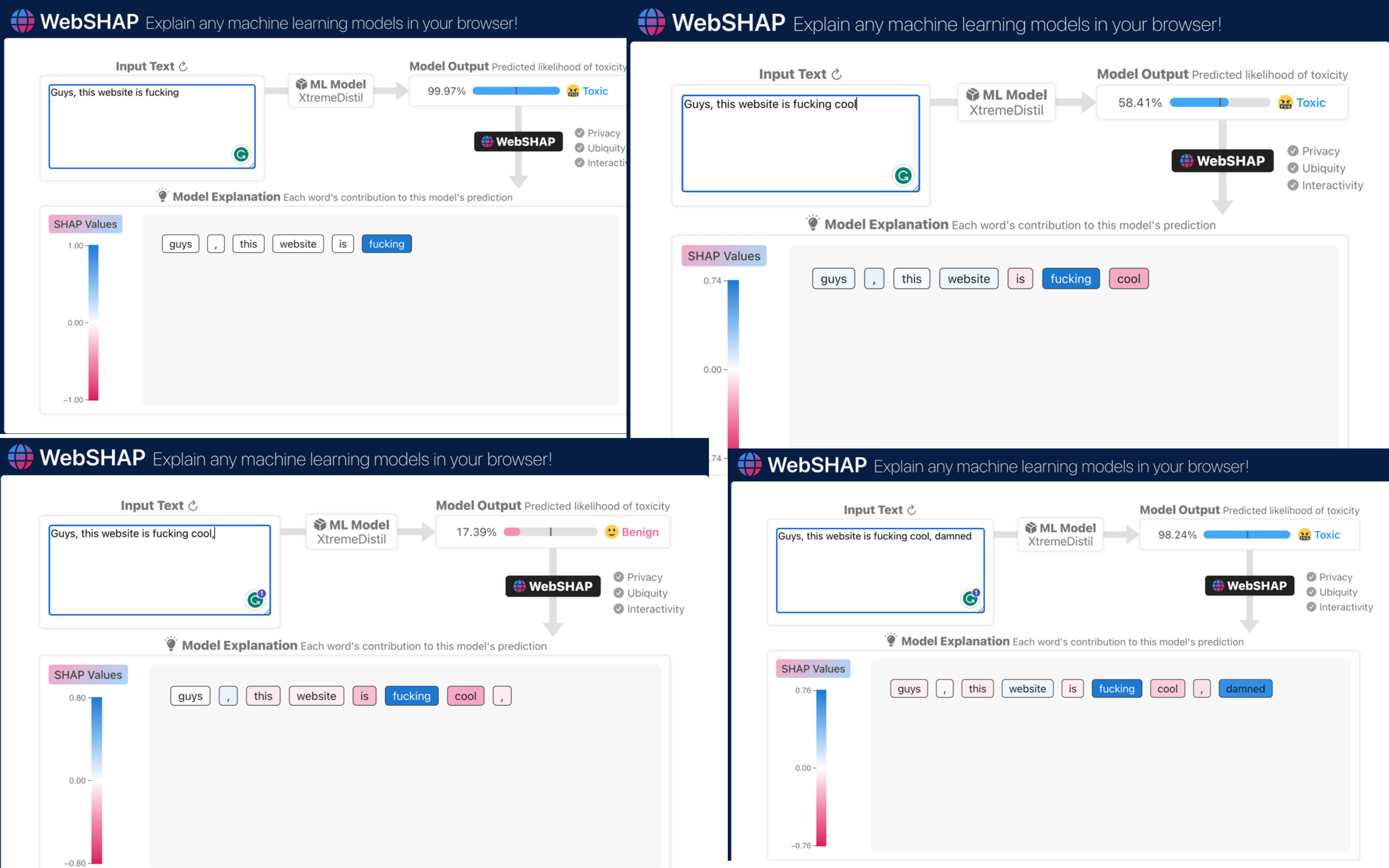
Properties of Shapley values

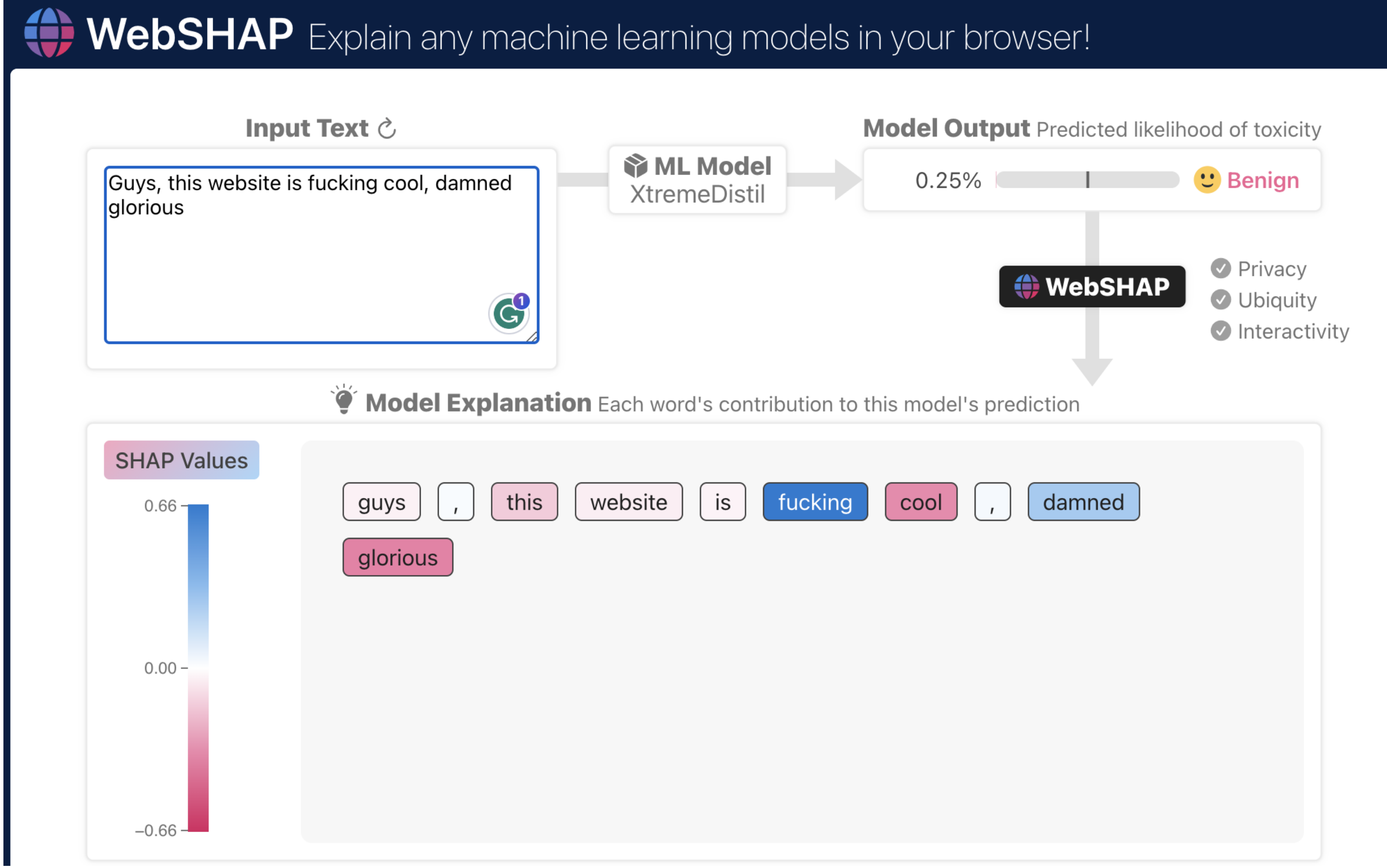
Efficiency Property.
Feature contributions add up to the difference of prediction from $x^\star$ and the expected value:

$$\sum_{j \in N} \mathcal{S}_j(f, x^\star) = f(x^\star) - E[f(X)]$$

Uninformativeness Property.
A feature j that does not change the predicted value has a Shapley value of zero.

$$\forall x, x_j, x_j' \colon f\left(\left\{x_{N \setminus \{j\}}, x_j\right\}\right) = f\left(\left\{x_{N \setminus \{j\}}, x_j'\right\}\right) \Rightarrow \forall x \colon \mathcal{S}_j(f, x) = 0$$


WebSHAP Explain any machine learning models in your browser!
Input Text
Guys, this website is fucking
ML Model XtremeDistil
Model Output Predicted likelihood of toxicity
99.97% Toxic
WebSHAP
Privacy
Ubiquity
Model Explanation Each word's contribution to this model's prediction
SHAP Values
1.00
0.00
−1.00
guys , this website is fucking
WebSHAP Explain any machine learning models in your browser!
Input Text
Guys, this website is fucking cool
ML Model XtremeDistil
Model Output Predicted likelihood of toxicity
58.41% Toxic
WebSHAP
Privacy
Ubiquity
Interactivity
Model Explanation Each word's contribution to this model's prediction
SHAP Values
0.74
0.00
guys , this website is fucking cool
WebSHAP Explain any machine learning models in your browser!
Input Text
Guys, this website is fucking cool,
ML Model XtremeDistil
Model Output Predicted likelihood of toxicity
17.39% Benign
WebSHAP
Privacy
Ubiquity
Interactivity
Model Explanation Each word's contribution to this model's prediction
SHAP Values
0.80
0.00
−0.80
guys , this website is fucking cool ,
WebSHAP Explain any machine learning models in your browser!
Input Text
Guys, this website is fucking cool, damned
ML Model XtremeDistil
Model Output Predicted likelihood of toxicity
98.24% Toxic
WebSHAP
Privacy
Ubiquity
Interactivity
Model Explanation Each word's contribution to this model's prediction
SHAP Values
0.76
0.00
−0.76
guys , this website is fucking cool , damned



- https://arxiv.org/abs/2303.09545

5 Decision Stumps

A Tree with Two Branches: Decision stump

- Training data $\{(x_i, y_i)\}_{i=1}^{N}$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1,1\}$
- A decision stump is a function $\phi_{j,s}$,
  which is parametrized by
  - polarity $p \in \{-1,1\}$
  - threshold $s \in \mathbb{R}$
  - index $j \in [1,2, \dots d]$

$$\phi_{j,s}(x_{\cdot}) = p\ \mathrm{sign}\left(x_{\cdot,j} - s\right) = \begin{cases} p, \text{if } x_{\cdot,j} \geq s \\ -p, \text{otherwise} \end{cases}$$

Select best decision stump

Error of decision stump

$$\widehat{Err}(\phi_{j,s}) = \frac{1}{N}\sum_{i=1}^{N}[\phi_{j,s}(x_i) \neq y_i] =$$

$$= \frac{1}{N}\sum_{i=1}^{N}[y_i(x_{i,j} - s) \leq 0]$$

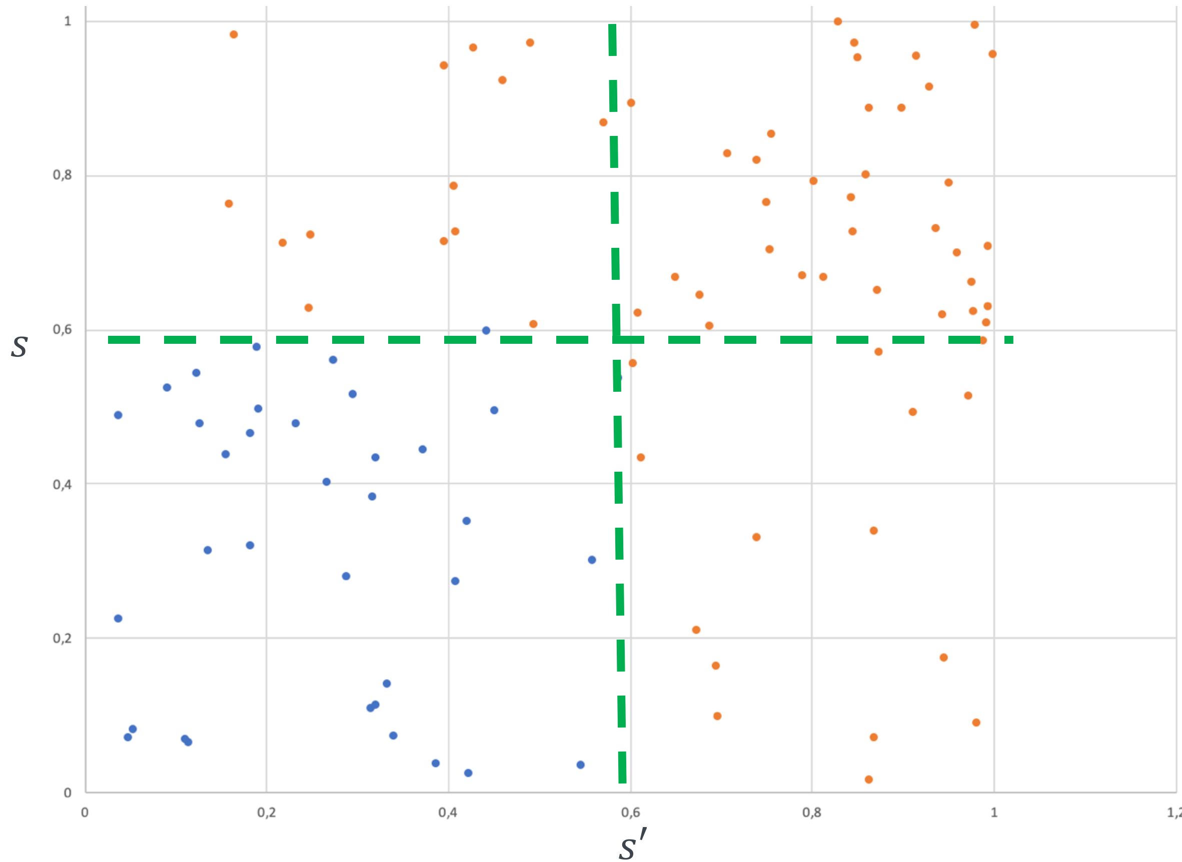
Determine

$$\underset{j\in\{1,\ldots,d\},s\in\{x_{i,j}|i\in\{1,\ldots,N\}\}}{\operatorname{argmin}} \widehat{Err}(\phi_{j,s})$$

Intuitively: a decision stump selector is a weak learner (with large training error)

Examples for decision stumps

$\phi_{2,s}(x) = \mathrm{sign}(x_{\cdot,2} - s)$ $\phi_{1,s'}(x) = \mathrm{sign}(x_{\cdot,1} - s')$



Other weak learners

- Selection from nominal features
- Selection from categorial features
- Decision trees
  - of limited depth

6 AdaBoost

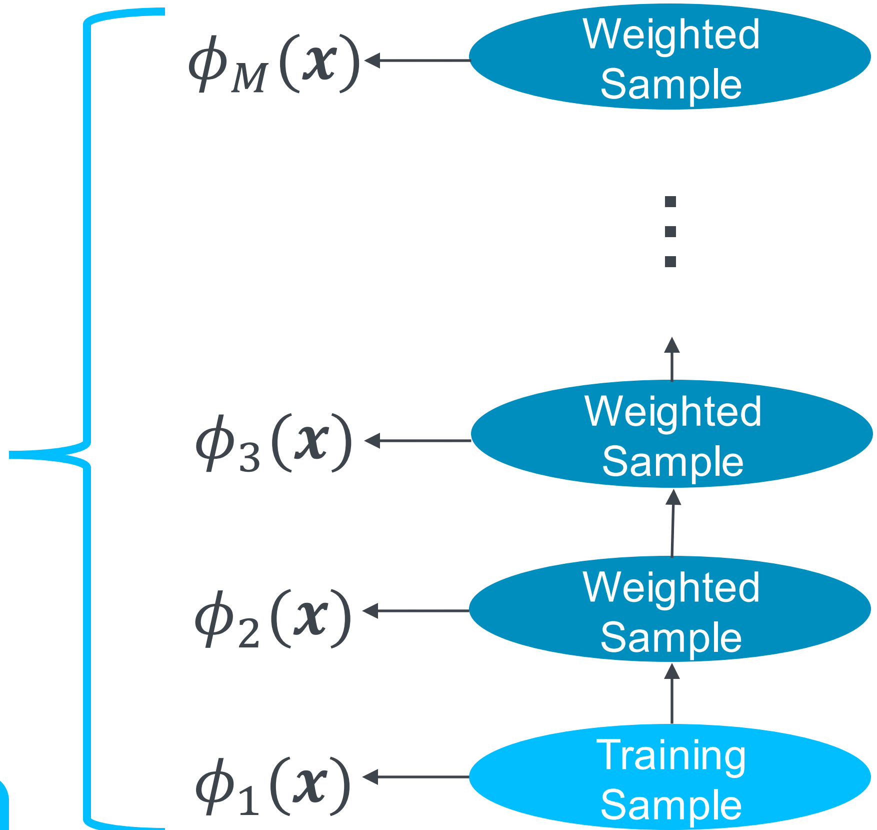
AdaBoost Scheme: Weighting data, weighting feature functions

- Assume infinite collection of feature functions $\phi_j: \mathbb{R}^d \to \{-1,1\}$
- Assume infinite vector $\theta = [\theta_1 \ \theta_2 \ldots]^T$ with finite number of non-zero entries
- Classifier:

$$\hat{f}_\theta(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{j=1}^{\infty} \theta_j \phi_j(\boldsymbol{x})\right)$$

- We also write $\sum_{j=1}^{\infty} \theta_j \phi_j(\boldsymbol{x}) = \theta^T \phi(\boldsymbol{x})$

How to select θ and $\phi(x)$?



Discrete AdaBoost Algorithm

1. **Init** weights $w_i = \frac{1}{N}$ for $i = 1 \dots N$
2. **Repeat**

   Could be decision stump $\phi_{j,s}(x)$

   a. Fit a classifier $\phi_m(x)$ to the training data using weights w_i

   b. Compute normalized weighted error

   $$\widehat{Err}(\phi_m) = \frac{\sum_{i=1}^{N} w_i\,[\phi_m(x_i) \neq y_i]}{\sum_{i=1}^{N} w_i}$$

   c. Compute log odds $\theta_m = \log \frac{1 - \widehat{Err}(\phi_m)}{\widehat{Err}(\phi_m)}$

   d. Set $w_i \coloneqq w_i \cdot e^{\theta_m [y_i \neq \phi_m(x_i)]}$ for $i = 1 \dots N$

   Until convergence of $\theta^T \phi(x)$
3. Output $\hat{f}(x) = \text{sign}[\theta^T \phi(x)]$

Side note:
„Real AdaBoost“ is a regression method

Illustration of AdaBoost

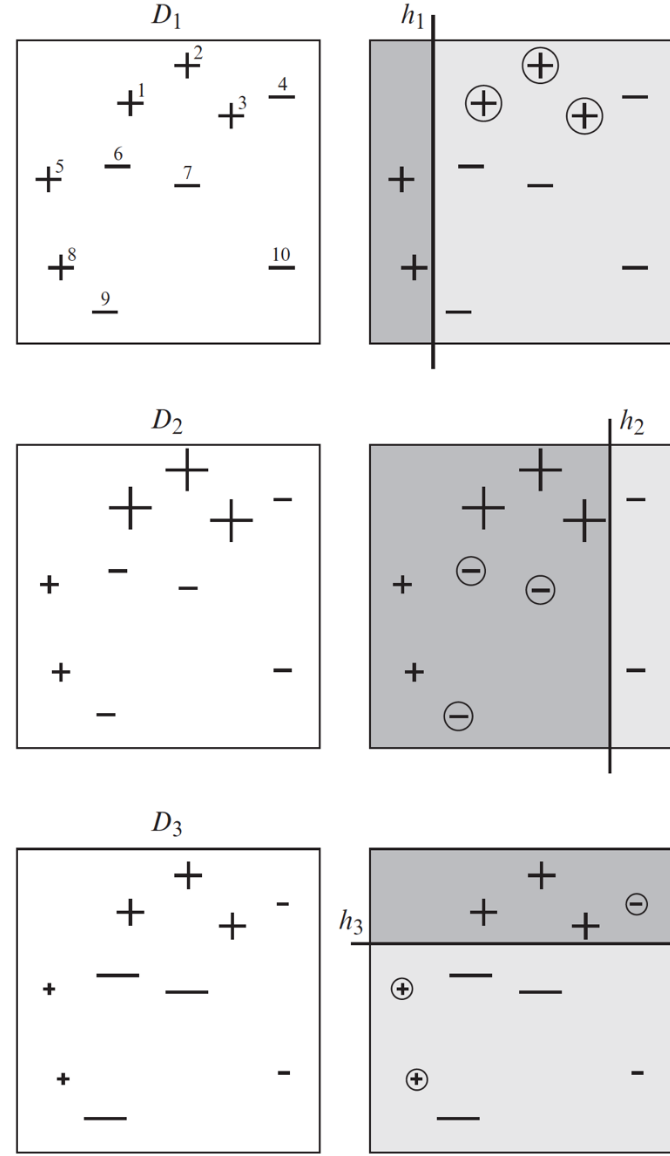
Each row depicts one round.

The left box in each row represents the training data, with the size of each example scaled in proportion to its weight.

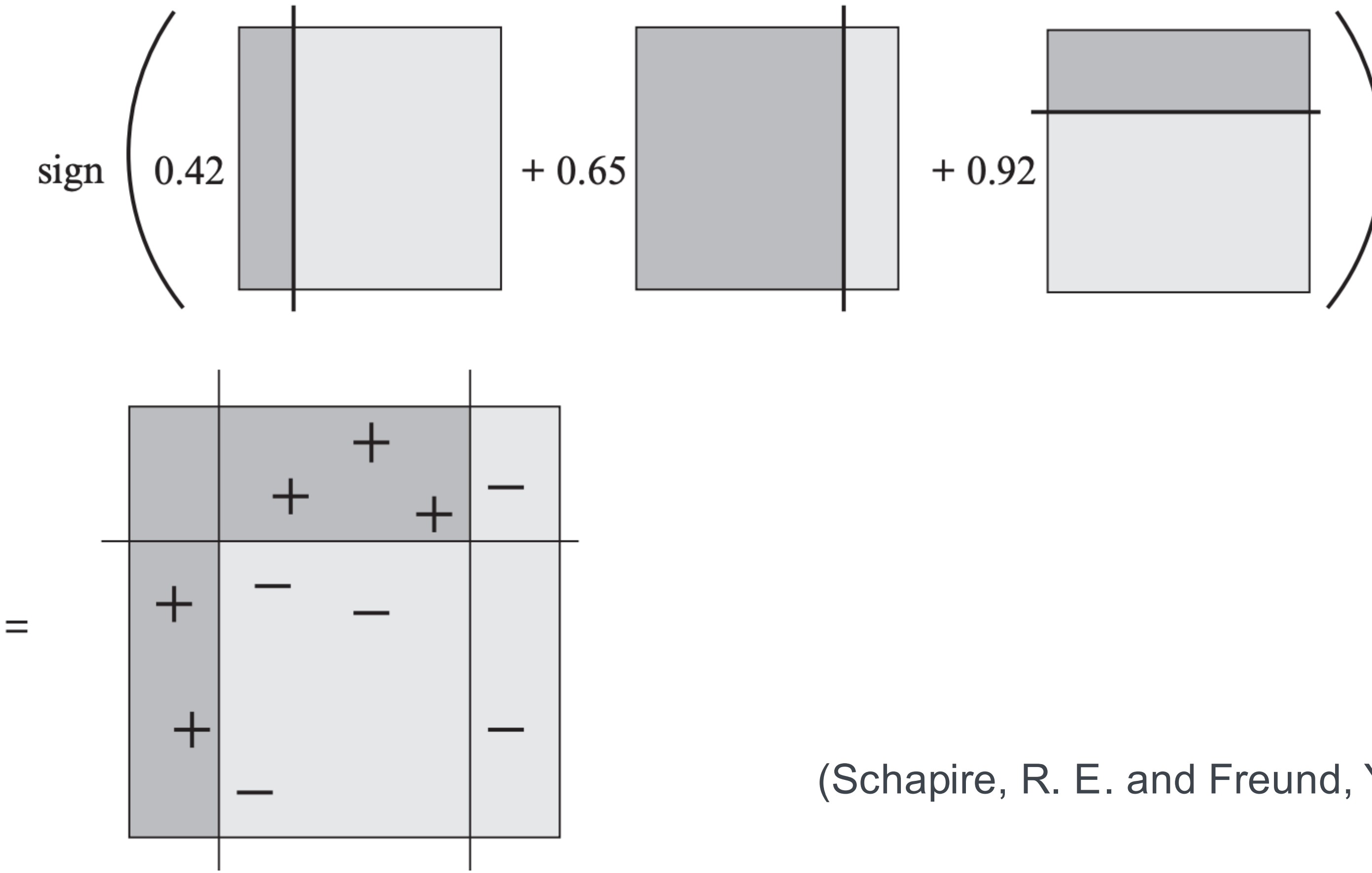
Each box on the right shows the weak hypothesis ϕ_m , where darker shading indicates the region of the domain predicted to be positive.

Examples that are misclassified by ϕ_m have been circled.

(Schapire, R. E. and Freund, Y. 2012), Figure 1.1.



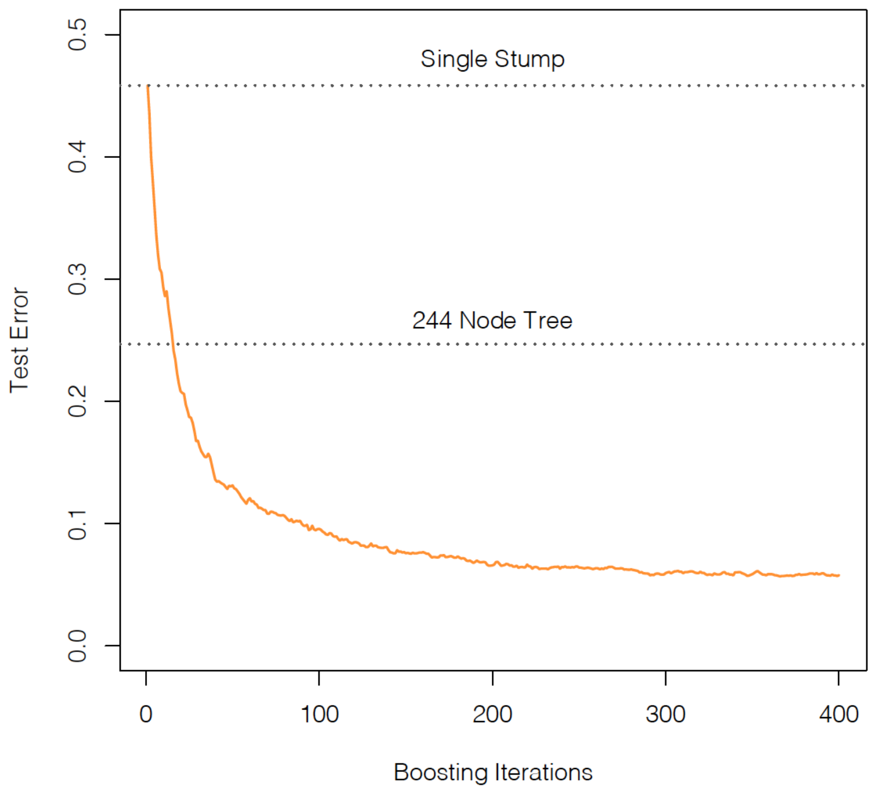
Combined ϕ_m



(Schapire, R. E. and Freund, Y. 2012), Figure 1.2.

Artificial Example: Sum of squares of 10 standard Gaussians

Artificial data $$Y = \begin{cases} 1, \text{if } \sum_{j=1}^{10} X_j^2 > \chi_{10}^2(0.5) = 9.34 \\ -1, \text{else} \end{cases}$$



- Single decision stump hardly better than random
- 400 combined stumps much better than a single large decision tree (error rate 24.7%)

(Hastie et al; Figure 10.2)

7 Understanding AdaBoost

Boosting Fits an Additive Model

Assume basis functions $b(x;\beta_m)$

$$\hat{f}(x) = \sum_{m=1}^{\infty} \theta_m b(x;\beta_m)$$

where

θ_m are the expansion coefficients and

β_m are the parameters

Side note: Additive expansions are at the core of many learning techniques
- single hidden-layer neural networks
- signal processing: wavelets or sinus functions as base functions
- multivariate adaptive regression splines

What should be avoided

- Do not aim at global optimization of loss

$$\xcancel{\underset{\{\theta_m,\beta_m\}_1^M}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{\infty} \theta_m b(x_i;\beta_m)\right)}$$

- Computationally too expensive
- Instead: Fit **one** basis function at a time!

Algorithm: Foreward Stagewise Additive Modeling

1. Initialize $\hat{f}_0(x) = 0$
2. Repeat until convergence
   a. Compute

$$(\theta_m, \beta_m) = \underset{\tilde{\theta}, \tilde{\beta}}{\operatorname{argmin}} \sum_{i=1}^{N} L\left(y_i, \hat{f}_{m-1}(x_i) + \tilde{\theta} b(x_i; \tilde{\beta})\right)$$

   b. Incrementally set

$$\hat{f}_m(x) = \hat{f}_{m-1}(x) + \theta_m b(x_i; \beta_m)$$

AdaBoost and Forward Stagewise Additive Modeling

- What is the connection between AdaBoost and Forward Stagewise Additive Modeling?
- Which loss function is used by AdaBoost?
- Exponential loss:

It took the community 5 years to find this result

$$L\left(y, \hat{f}(x)\right) = e^{-y\hat{f}(x)}$$

- Solve

$$(\theta_m, \phi_m) = \underset{\tilde{\theta},\tilde{\phi}}{\operatorname{argmin}} \sum_{i=1}^{N} e^{-y_i\left(\hat{f}_{m-1}(x_i)+\tilde{\theta}\tilde{\phi}(x_i)\right)}$$

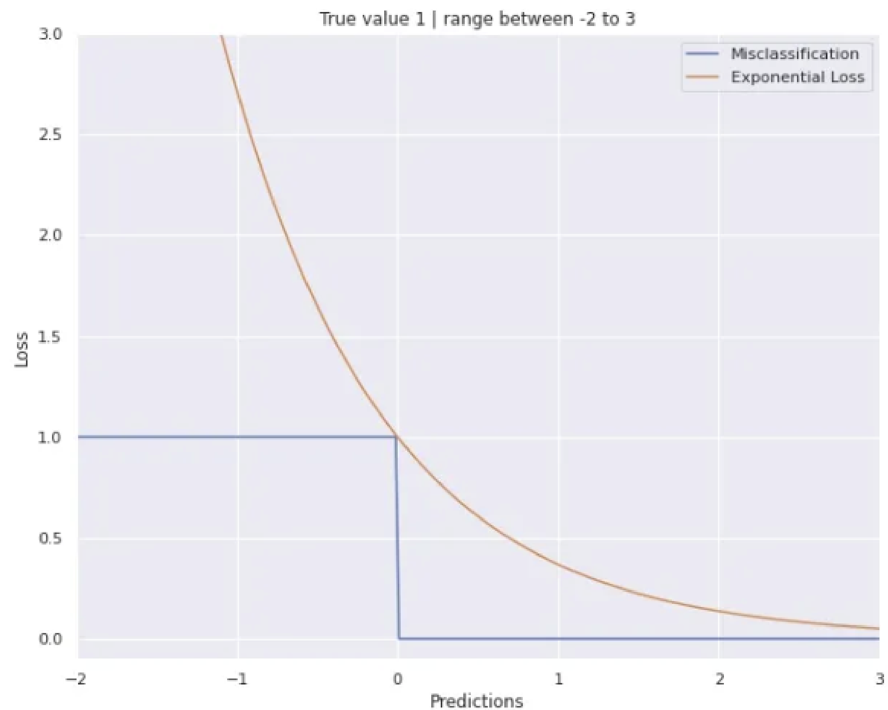
Exponential loss

vs

0-1-loss

```
def exponential_loss(y_pred, y_true):
    return np.mean(np.exp(- y_pred * y_true))
```

The result can be shown below:



Exponential Loss vs misclassification (1 if y<0 else 0)

https://towardsdatascience.com/what-is-loss-function-1e2605aeb904

AdaBoost and Forward Stagewise Additive Modeling

- Exponential loss: $L\left(y, \hat{f}(x)\right) = e^{-y\hat{f}(x)}$
- Solve

$$(\theta_m, \phi_m) = \underset{\tilde{\theta},\tilde{\phi}}{\operatorname{argmin}} \sum_{i=1}^{N} e^{-y_i\left(\hat{f}_{m-1}(x_i) + \tilde{\theta}\tilde{\phi}(x_i)\right)} =$$

$$= \underset{\tilde{\theta},\tilde{\phi}}{\operatorname{argmin}} \sum_{i=1}^{N} e^{-y_i \hat{f}_{m-1}(x_i)} \, e^{-y_i\tilde{\theta}\tilde{\phi}(x_i)} = \underset{\tilde{\theta},\tilde{\phi}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} \, e^{-y_i\tilde{\theta}\tilde{\phi}(x_i)}$$

remember the definition of weights later!

- For any $\tilde{\theta} \geq 0$ this is equivalent to:

$$\underset{\tilde{\theta},\tilde{\phi}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} \left[y_i \neq \tilde{\phi}(x_i)\right]$$

because…

Proving the claim: Relating exponential loss and weighted error

$$\mathbf{\underset{\tilde{\theta},\tilde{\phi}}{argmin} \sum_{i=1}^{N} w_i^{(m)}\, e^{-y_i \tilde{\theta} \tilde{\phi}(x_i)} \overset{?}{\underset{!}{=}} \underset{\tilde{\theta},\tilde{\phi}}{argmin} \sum_{i=1}^{N} w_i^{(m)}\, [y_i \neq \tilde{\phi}(x_i)]}$$

because

$$\underset{\tilde{\theta},\tilde{\phi}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)}\, e^{-y_i \tilde{\theta} \tilde{\phi}(x_i)} =$$

$$= \underset{\tilde{\theta},\tilde{\phi}}{\operatorname{argmin}}\, e^{-\tilde{\theta}} \sum_{y_i = \tilde{\phi}(x_i)} w_i^{(m)} + e^{\tilde{\theta}} \sum_{y_i \neq \tilde{\phi}(x_i)} w_i^{(m)} =$$

Consider all combinations of exponents and true/false predictions

$$= \underset{\tilde{\theta},\tilde{\phi}}{\operatorname{argmin}} \left(e^{\tilde{\theta}} - e^{-\tilde{\theta}}\right) \sum_{i=1}^{N} w_i^{(m)}\, [y_i \neq \tilde{\phi}(x_i)] + e^{-\tilde{\theta}} \sum_{i=1}^{N} w_i^{(m)} =$$

division by positive constant does not change argmin

$$= \underset{\tilde{\theta},\tilde{\phi}}{\operatorname{argmin}} \left(e^{\tilde{\theta}} - e^{-\tilde{\theta}}\right) \frac{\sum_{i=1}^{N} w_i^{(m)}\, [y_i \neq \tilde{\phi}(x_i)]}{\sum_{i=1}^{N} w_i^{(m)}} + e^{-\tilde{\theta}} =$$

plug-in weighted normalized error rate

$$= \underset{\tilde{\theta},\tilde{\phi}}{\operatorname{argmin}} \left(e^{\tilde{\theta}} - e^{-\tilde{\theta}}\right) \widehat{Err}(\tilde{\phi}) + e^{-\tilde{\theta}}$$

Proving the claim: $\tilde{\theta} = \frac{1}{2}\log\frac{\left(1-\widehat{Err}(\tilde{\phi})\right)}{\widehat{Err}(\tilde{\phi})}$

$$\frac{\boldsymbol{\partial\left(e^{\tilde{\theta}} - e^{-\tilde{\theta}}\right)\widehat{Err}\left(\tilde{\phi}\right) + e^{-\tilde{\theta}}}}{\boldsymbol{\partial\tilde{\theta}}} = (e^{\tilde{\theta}}+e^{-\tilde{\theta}})\widehat{Err}\left(\tilde{\phi}\right) - e^{-\tilde{\theta}} = \mathbf{0}$$

$$\widehat{Err}\left(\tilde{\phi}\right)e^{\tilde{\theta}} - \left(1 - \widehat{Err}\left(\tilde{\phi}\right)\right)e^{-\tilde{\theta}} = 0$$

$$\widehat{Err}\left(\tilde{\phi}\right)e^{\tilde{\theta}} = \left(1 - \widehat{Err}\left(\tilde{\phi}\right)\right)e^{-\tilde{\theta}}$$

$$\tilde{\theta} + \log\widehat{Err}\left(\tilde{\phi}\right) = -\tilde{\theta} + \log\left(1 - \widehat{Err}\left(\tilde{\phi}\right)\right)$$

$$2\tilde{\theta} = \log\left(1 - \widehat{Err}\left(\tilde{\phi}\right)\right) - \log\widehat{Err}\left(\tilde{\phi}\right) = \log\frac{\left(1 - \widehat{Err}\left(\tilde{\phi}\right)\right)}{\widehat{Err}\left(\tilde{\phi}\right)}$$

2a Updating in Forward Stagewise Additive Modeling

$$\hat{f}_m(x) = \hat{f}_{m-1}(x) + \theta_m \boldsymbol{\phi}_m(x)$$

Causing next weights to be

$$w_i^{(m+1)} = \underbrace{w_i^{(m)} \cdot e^{-y_i \theta_m \boldsymbol{\phi}_m(x_i)}}_{\text{use update like this}} = \underbrace{w_i^{(m)} \cdot e^{\alpha_m [y_i \neq \boldsymbol{\phi}_m(x_i)]} \cdot e^{-\theta_m}}_{\text{or bring it into form used by AdaBoost 2d,}}$$

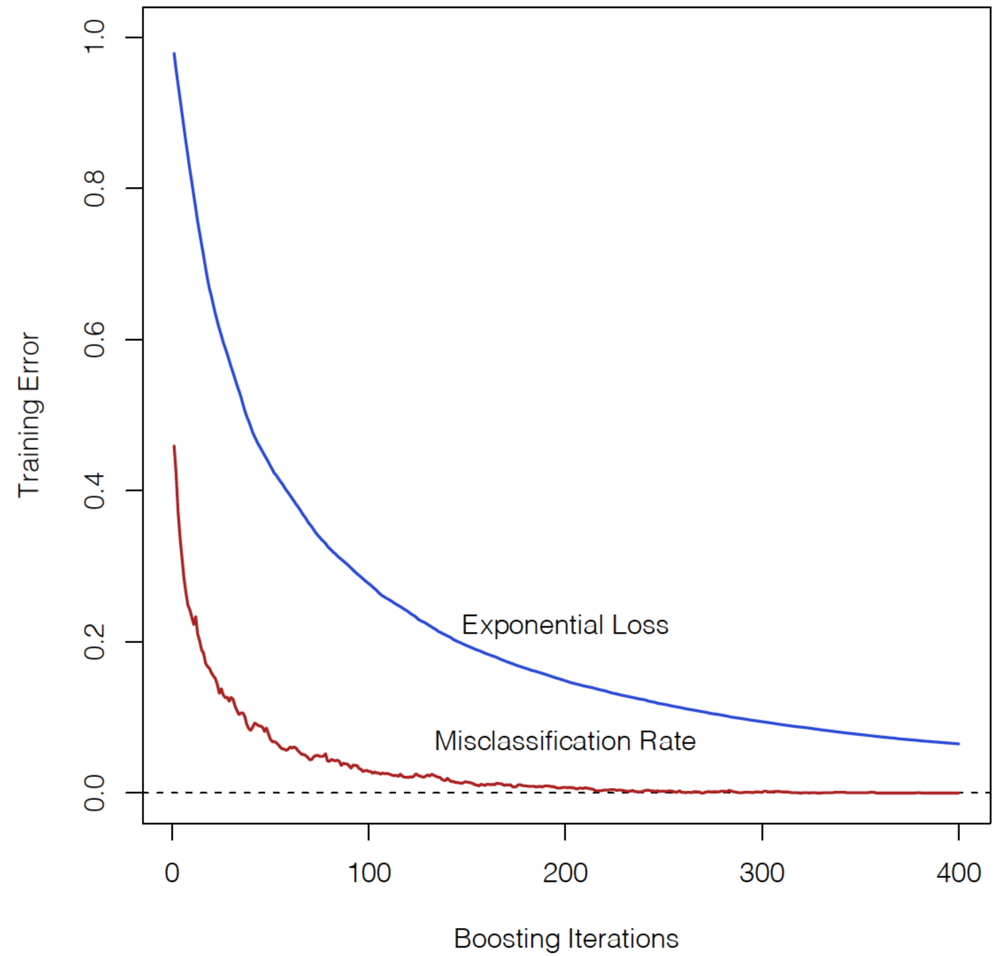
with $\alpha_m = 2\theta_m$

and $e^{-\theta_m}$ a constant factor to all weights that is therefore ignored

8 Understanding Losses

Artificial Example: Sum of squares of 10 standard Gaussians

- After 250 iterations the training error is 0,
- but exponential loss continues to decrease



(Hastie et al), Figure 10.3

Comparing loss functions for binary classification

- Missclassification:

$$[sign(\hat{f}) \neq y]$$

- Exponential:

$$e^{-y\hat{f}}$$

- Cross entropy (binomial deviance):

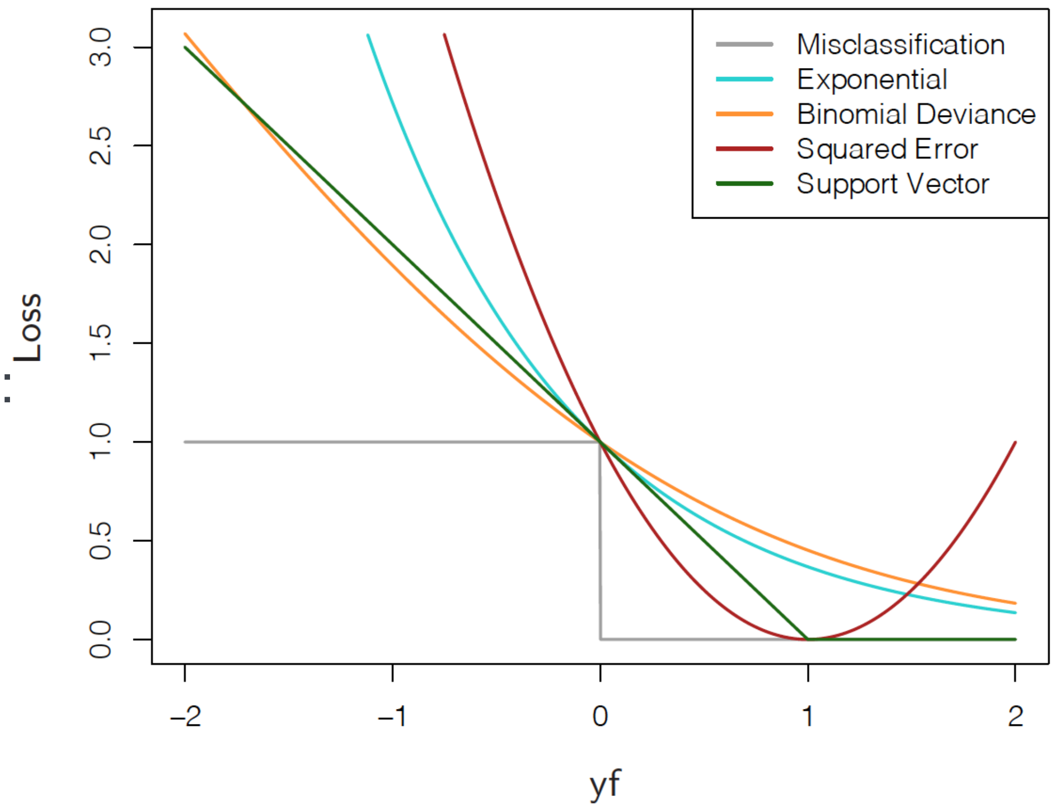
$$\log\left(1 + e^{-2y\hat{f}}\right)$$

- Squared error:

$$\left(y - \hat{f}\right)^2$$

- Hinge loss (support vector):

$$\max(0,1 - y\hat{f})$$



Comparing loss functions for binary classification

- Squared error loss does not decrease continuously,

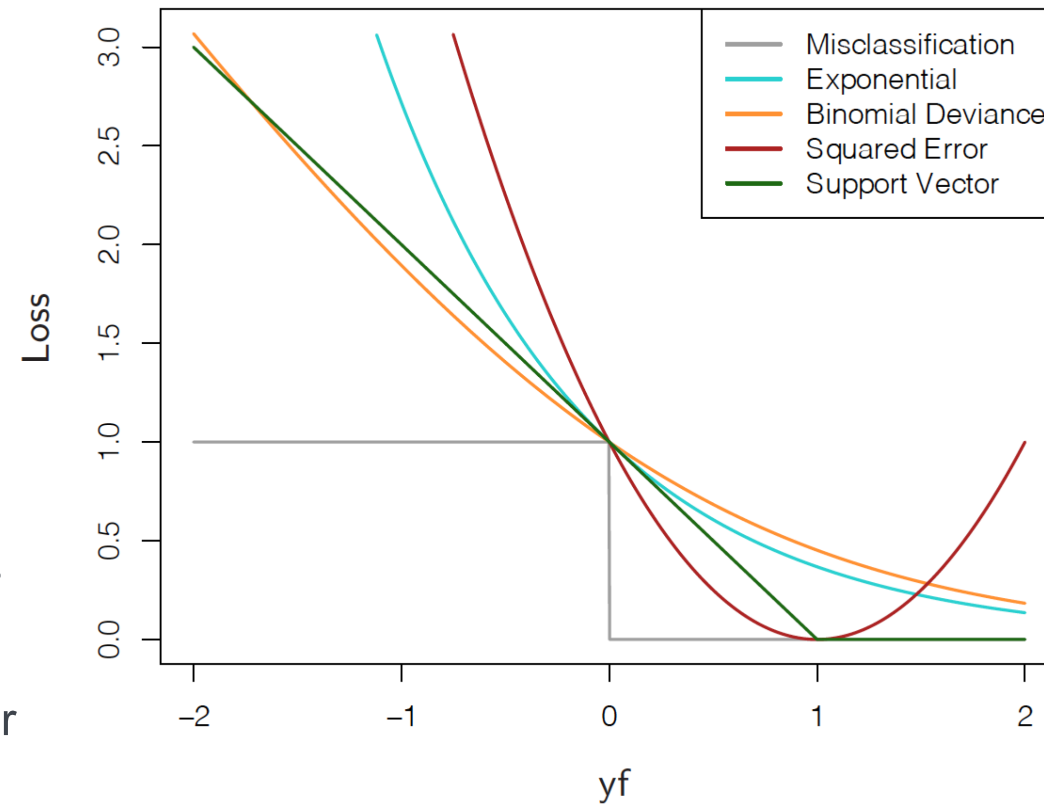
$\Longrightarrow$ unnecessary attention on correctly classified entities

- Exponential loss punishes wrong predictions exponentially

$\Longrightarrow$ problematic when Bayes error is high

$\Longrightarrow$ AdaBoost deteriorates in such settings

- Cross entropy and hinge loss deal better with high Bayes error in the data set



Comparing loss functions for regression

- Squared error:

$$\left(y - \hat{f}\right)^2$$

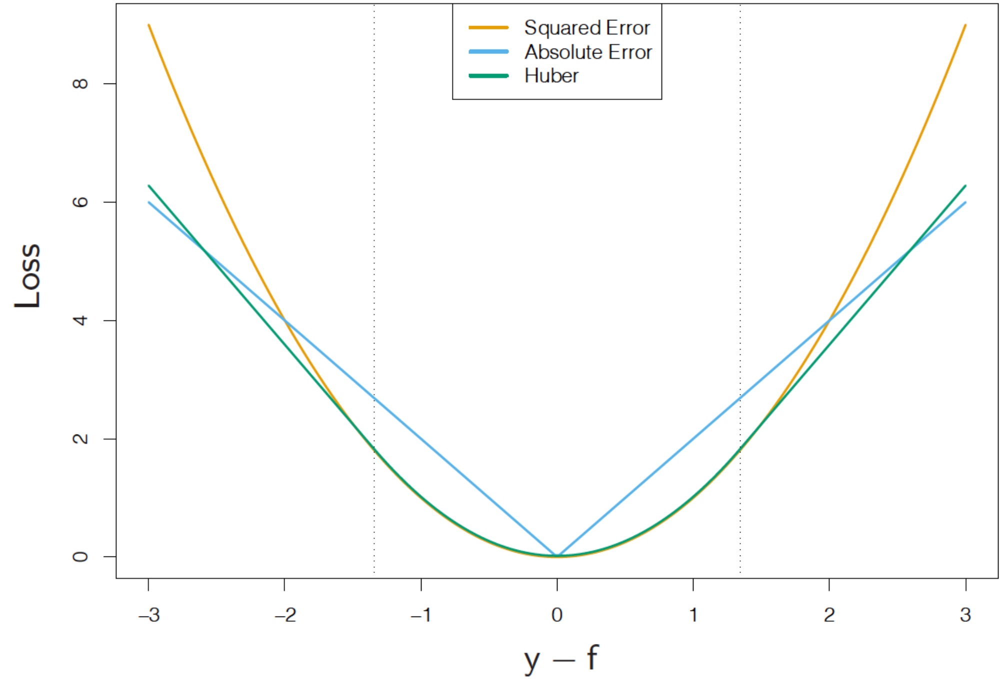
- Absolute error

$$|y - \hat{f}|$$

- Huber

$$L\left(y, \hat{f}(x)\right) =$$

$$= \begin{cases} \left[y - \hat{f}(x)\right]^2, & \text{for } |y - \hat{f}(x)| < \delta \\ 2\delta\left|y - \hat{f}(x)\right| - \delta^2, & \text{otherwise} \end{cases}$$

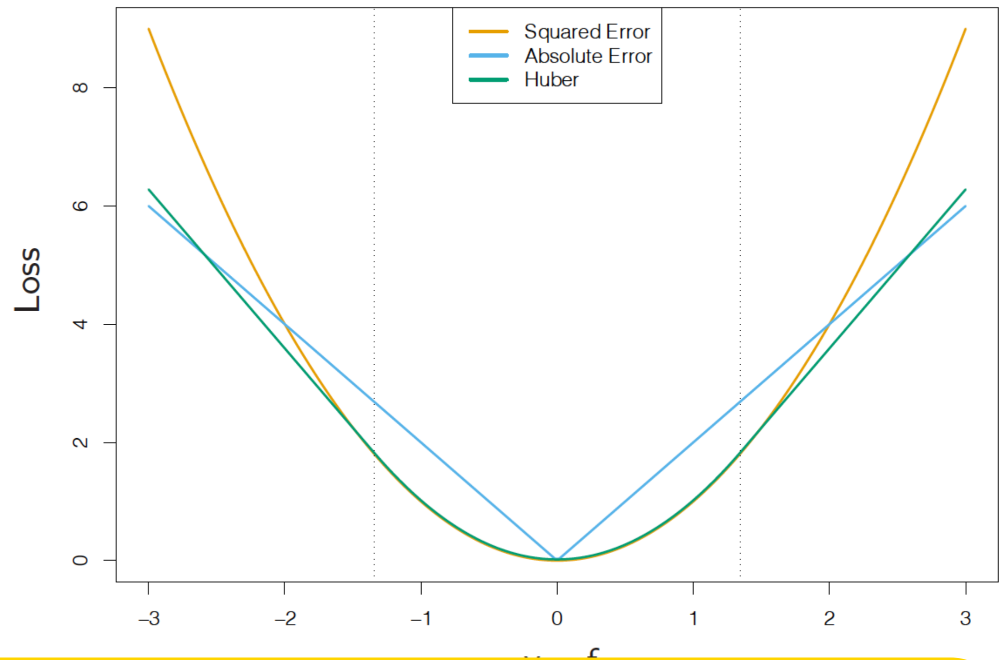


Comparing loss functions for regression

- Squared error loss emphasizes large absolute residuals $|y - \hat{f}|$

$\Longrightarrow$ less robust against outliers

$\Longrightarrow$ problematic with long-tailed error distributions



Some loss functions perform badly in certain situations,

- Example *classification*: Squared error loss and exponential loss for boosting
- Example *regression*: Squared error loss for linear regression

but they lead to simple and elegant methods, which others don‘t

Universität Stuttgart
KI

Thank you!

Steffen Staab

E-Mail Steffen.staab@ki.uni-stuttgart.de
Telefon +49 (0) 711 685-88100
www.ki.uni-stuttgart.de/

Universität Stuttgart
Analytic Computing, KI
Universitätsstraße 32, 50569 Stuttgart