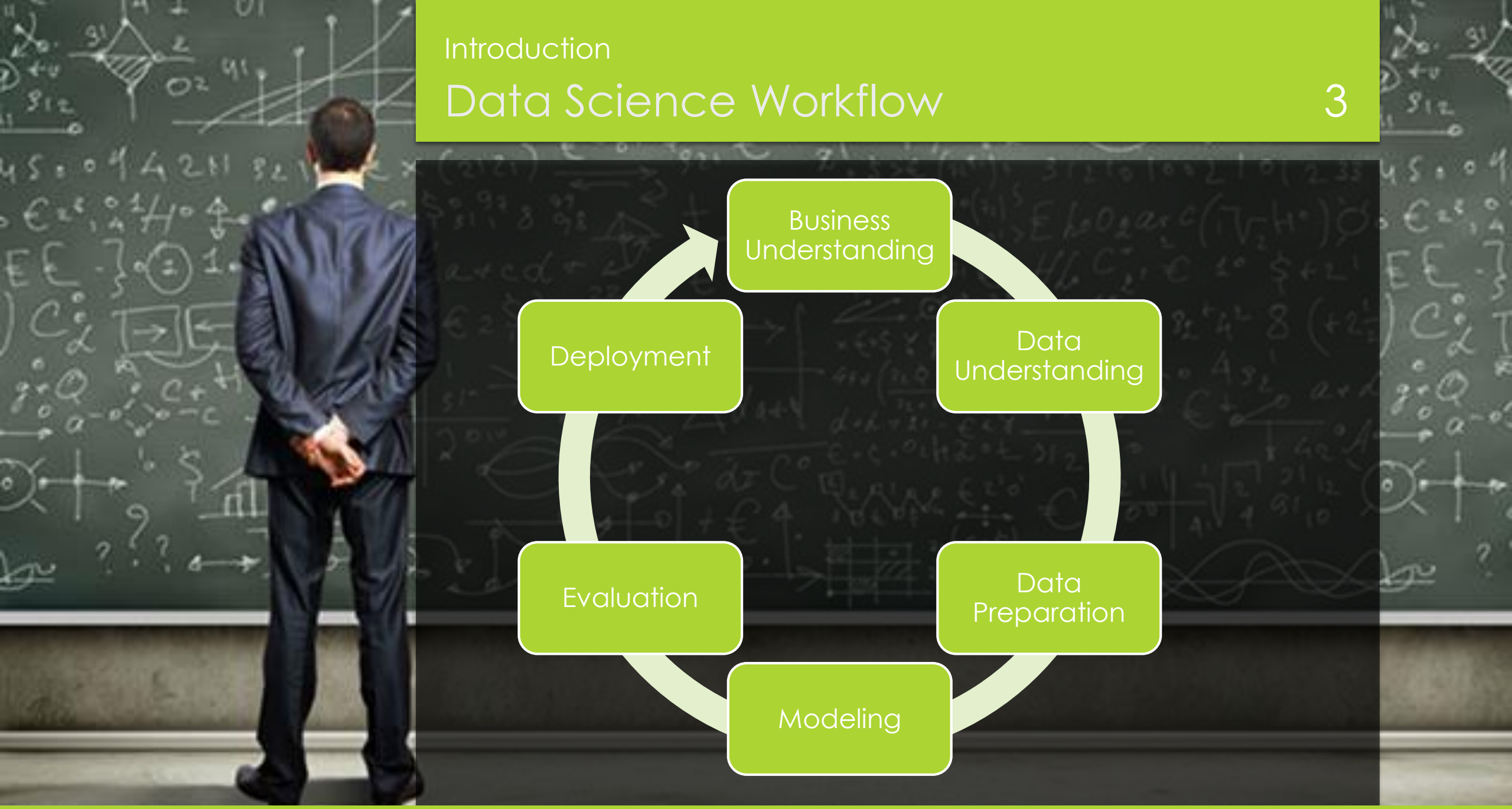
Automatic Gradient Boosting

INTRODUCING THE AUTOXGBOOST R-PACKAGE

Contents

Data Science Workflow



Modeling by AutoML

Idea:

- Simplify Analysis

How?

- Make modelbuilding parameter-free for user

Combined Algorithm Selection and Hyperparameter optimization (CASH)

Motivation:

- no single machine learning method performs best on all datasets
- machine learning algorithms mostly rely on hyperparameter optimization to perform well

Examples

- Auto-WEKA (786-dimensions)
- auto-sklearn (110-dimensions)
- H2O AutoML
- TPOT
- pennAI
- AutoCompete

Single-Learner Approaches

Motivation:

- Single machine learning algorithm might be sufficient for most problems

  → make it parameter-free including i.a.

  - Data preprocessing
  - Hyperparameter tuning

Examples

- autoxgboost
- Tuneranger
- Driverless AI
- PiSTOL

Tuning frameworks:

- RoBo

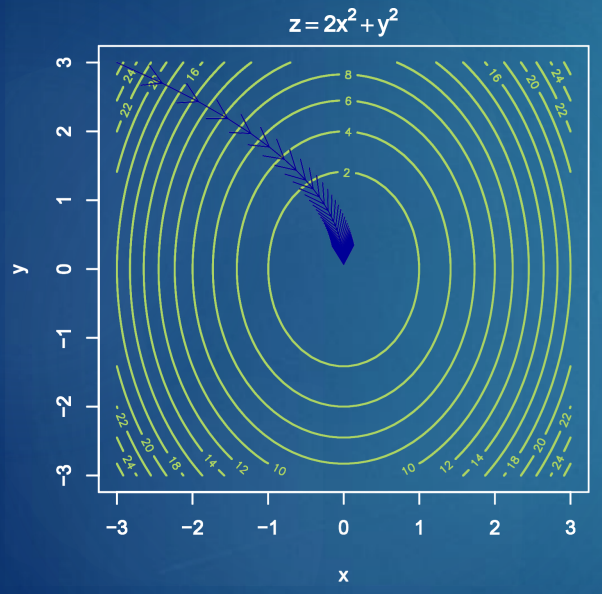
autoxgboost – an Automatic Gradient Boosting Machine

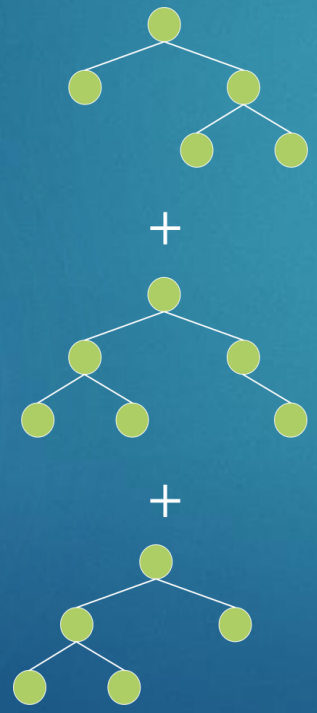
Gadient Boosting

= Gradient Descent + Tree Boosting

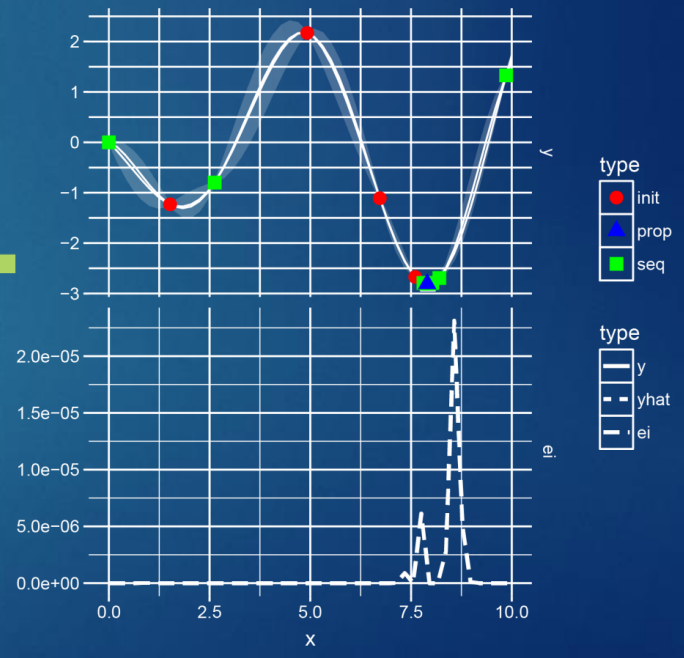
Factor Encoding

Hyperparameter Tuning





X	X^*
a	1
b	2
a	1
c	3
b	2
c	3
a	1
c	3
b	2



Gradient Descent

Optimization method

- Deterministic
- nonparametric
- Iterative

Consider arbitrary, differentiable target function $f(x)$, then

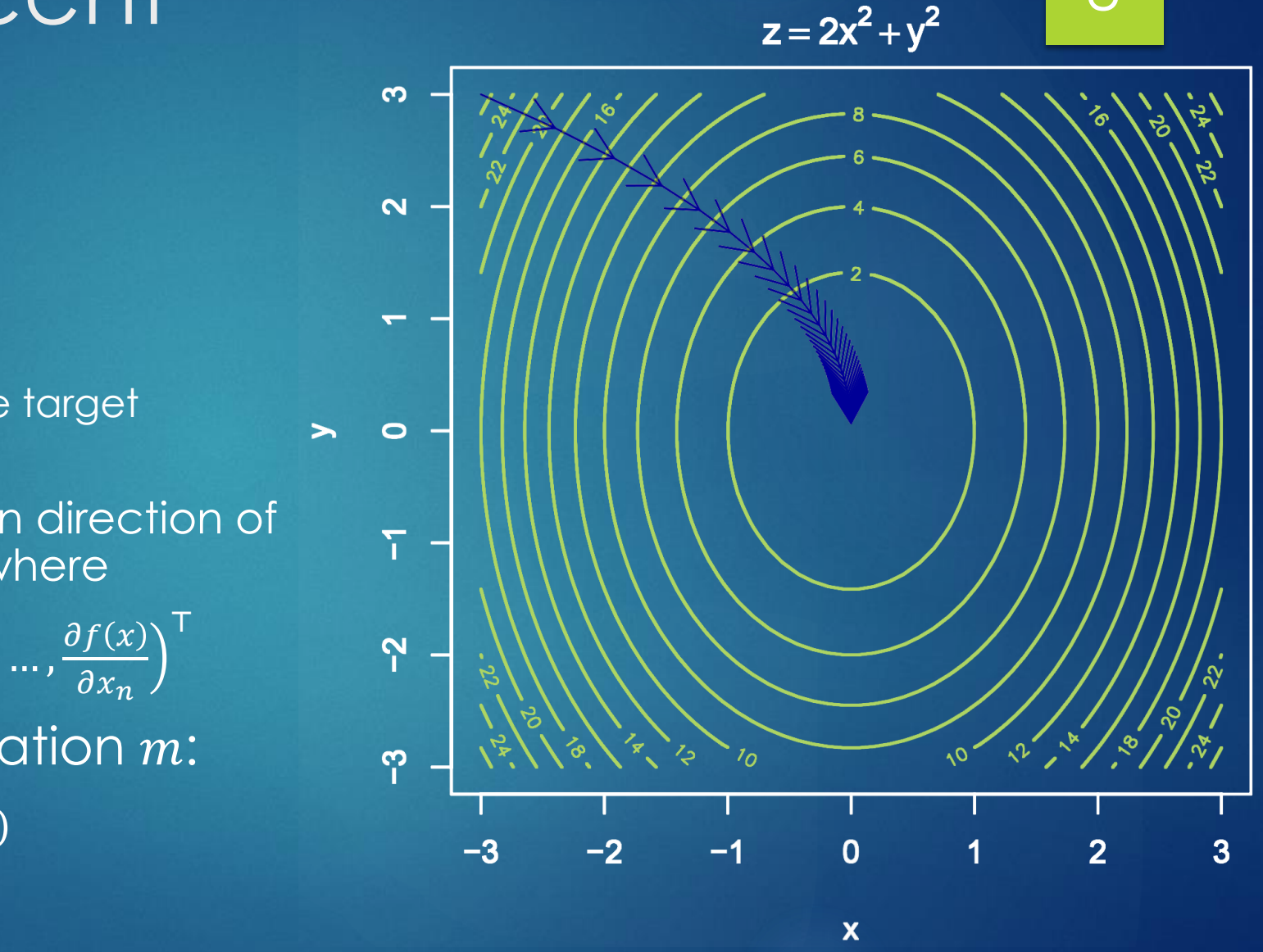
- neg. gradient $-\nabla f(x)$ points in direction of steepest descent of $f(x)$, where

$$\nabla f(x) = grad\ f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n}\right)^T$$

Optimization step for iteration m:

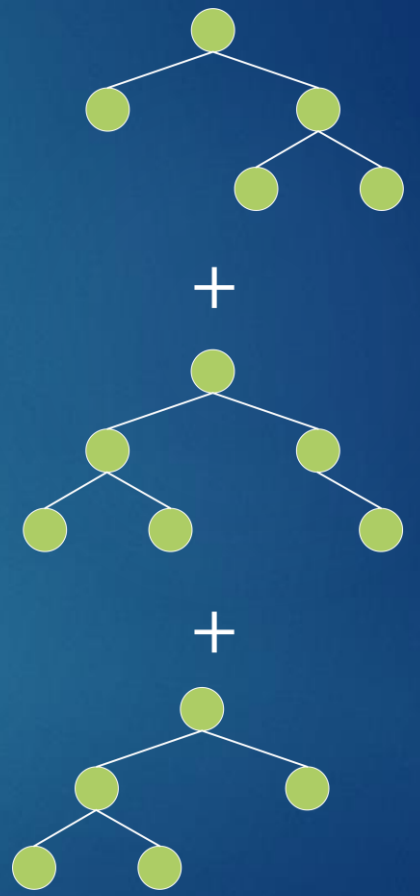
$$x^{(m)} = x^{(m-1)} - \nu \nabla f(x^{(m-1)})$$

With ν as stepsize



Boosting
Stagewise Additive Modeling

- Schapire et al. (1998) showed that the performance of a weak learner could always be improved by training additional ones



Additive Model:

$$f(x) = \sum_{m=1}^{M} f_m(x) = \sum_{m=1}^{M} \beta_m h(x, \theta_m)$$

with weak learner $h(x, \theta)$ (e.g. tree stumps).

Goal: Minimize empirical risk R in each iteration m

for parameters (β_m, θ_m), s. t.

$$(\beta_m^*, \theta_m^*) = \underset{\beta,\theta}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \beta h(x_i, \theta)$$

to get update $f_m(x) \leftarrow f_{m-1}(x) + \beta_m h(x_i, \theta_m)$

Gradient Tree Boosting (Regression)

- Look at negative gradient of the epirical risk at point x_i and iteration m:

$$-\nabla R|_{x_i,m} = \frac{\partial L(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)} =: r_{im}$$ Pseudo Residuals

- Those Pseudo Residuals can be approximated by regression tree (weak learner)

$h(x, \theta_m) = h(x, b, R) = \sum_{j=1}^{n} b_j \, \mathrm{I}(x \in R_j)$, with means b_j on terminal partitions R_j.

- Hence, an update for iteration m is given by

$$f_m(x) \leftarrow f_{m-1}(x) - \sum_j^{J_m} \gamma_{jm} I(x \in R_{jm}), \quad \text{with } \gamma_{jm} = \beta_m b_{jm}$$

whose optimal param γ_{jm} can be found by minimizing loss

$$\underset{\gamma}{\operatorname{argmin}} \sum_{x \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

Gradient Boosting implementations

- eXtreme Gradient Boosting (XGBoost)
  - State-of-the-art Gradient Boosting implementation in mlr
  - Base for autoxgboost
  - Introduces regularization to the objective function (Lasso/Ridge combination)
  - Fast and native support of parallel computing
  - Keeps used ressources to a minimum
- mboost
  - model-based boosting, i.e. glm, gam, trees
- GBM
- H2O-GBM
- lightGBM
  - leaf wise instead of level wise tree building → fast!
- CatBoost
  - Supports categorical features

} frequently outperform XGBoost

Factor Encoding

Integer | **Dummy** | **Impact**

X
a
b
a
c
b
c
a
c
b

→

X^*
1
2
1
3
2
3
1
3
2

X
a
b
a
c
b
c
a
c
b

→

X_a^*	X_b^*	X_c^*
1	0	0
0	1	0
1	0	0
0	0	1
0	1	0
0	0	1
1	0	0
0	0	1
0	1	0

X	Y
a	1
b	1
a	1
c	0
b	1
c	0
a	1
c	1
b	0

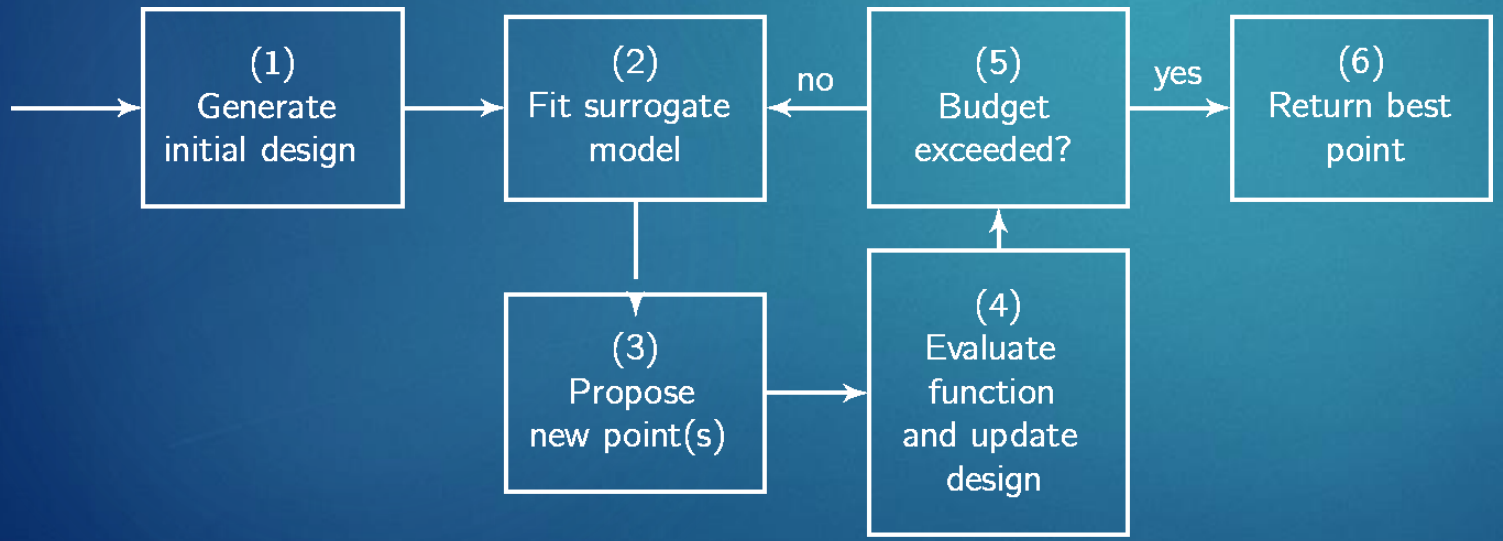
→

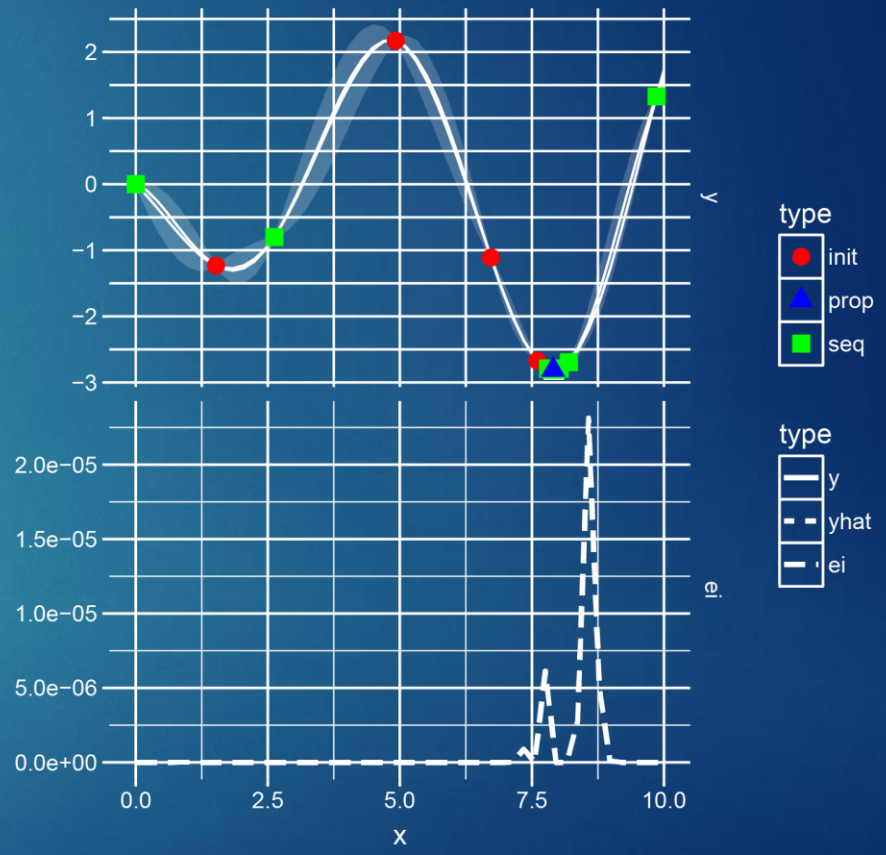
X^*	Y
$\lambda(n_a)P(Y=1\|X=a)+(1-\lambda(n_a)P(Y=1)$	1
$\lambda(n_b)P(Y=1\|X=b)+(1-\lambda(n_b)P(Y=1)$	1
$\lambda(n_a)P(Y=1\|X=a)+(1-\lambda(n_a)P(Y=1)$	1
$\lambda(n_c)P(Y=1\|X=c)+(1-\lambda(n_c)P(Y=1)$	0
$\lambda(n_b)P(Y=1\|X=b)+(1-\lambda(n_b)P(Y=1)$	1
$\lambda(n_c)P(Y=1\|X=c)+(1-\lambda(n_c)P(Y=1)$	0
$\lambda(n_a)P(Y=1\|X=a)+(1-\lambda(n_a)P(Y=1)$	1
$\lambda(n_c)P(Y=1\|X=c)+(1-\lambda(n_c)P(Y=1)$	1
$\lambda(n_b)P(Y=1\|X=b)+(1-\lambda(n_b)P(Y=1)$	0

Hyperparameter tuning

with mlrMBO

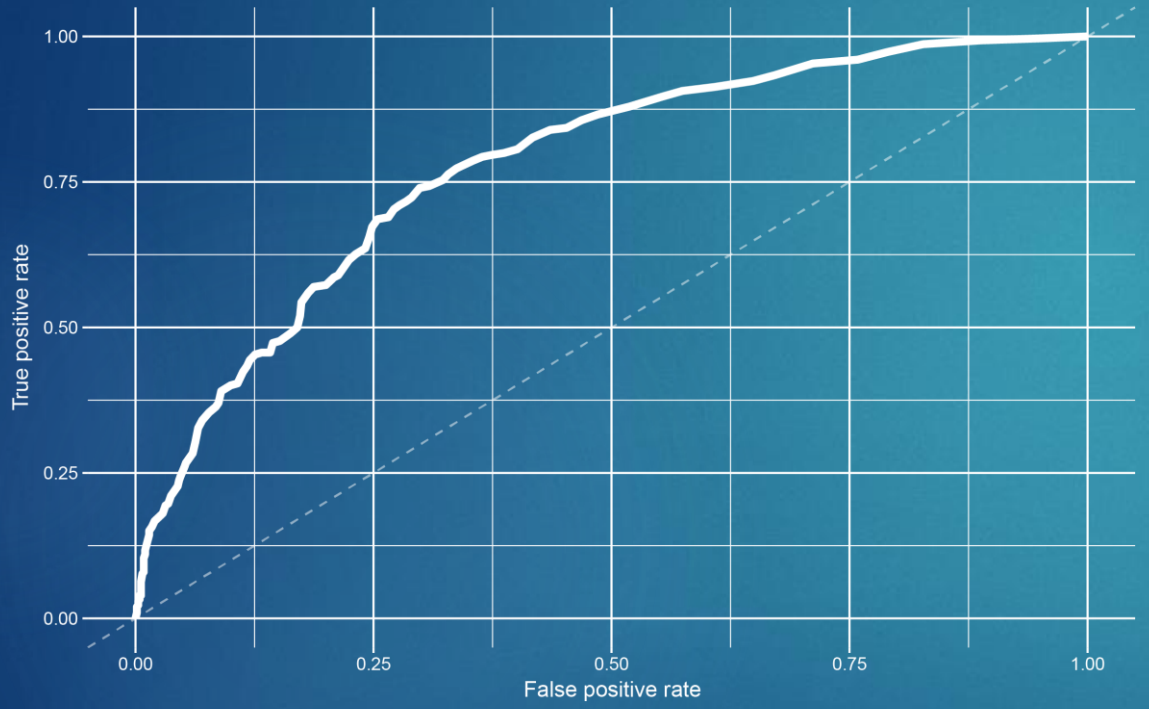
- Based on Sequential Model-Based Optimization (SMBO)
- Uses Gaussian Processes as surrogate models for function approximations
- New points are proposed depending on their expected improvement



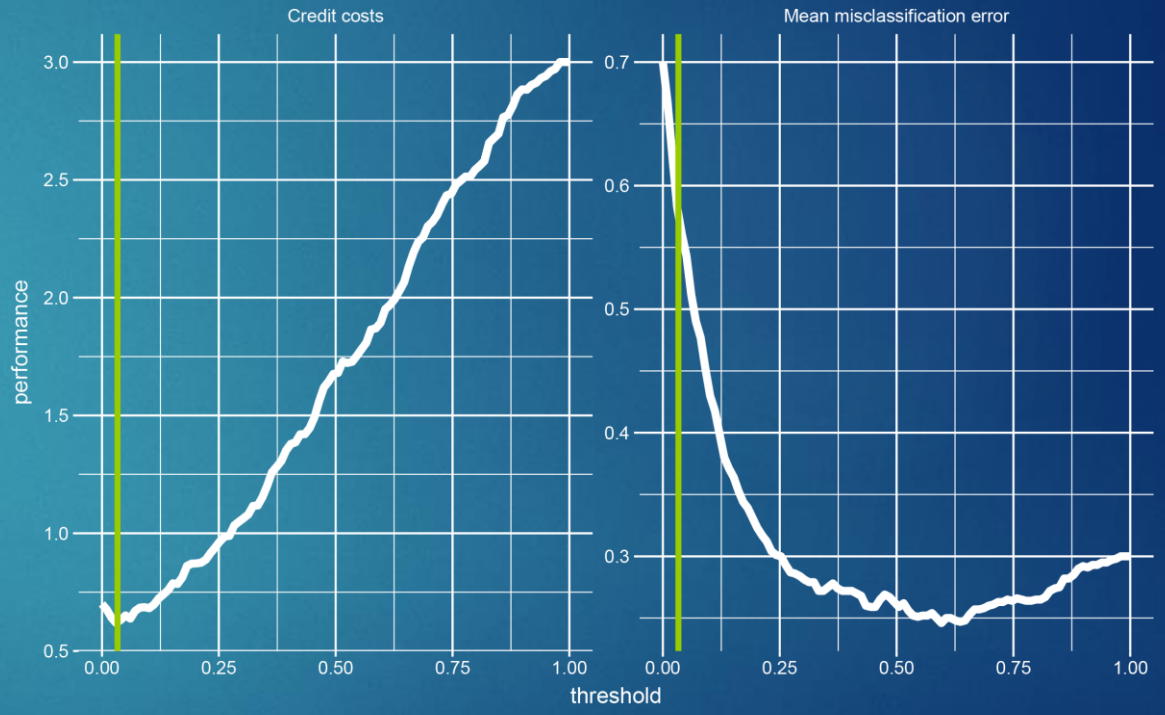


Threshold Tuning for Classification

Binary Classification Example: Credit Costs



ROC curve



Tuned threshold for cost-sensitive classification

Threshold Tuning for Classification

Details

- Probabilities returned by a model not always trustworthy, e. g.
  - in areas of high uncertainty (e.g. around 0.5 for binary classification)
  - for highly unbalanced datasets
  - for measurement errors
  - cost-sensitive classification
- Concept is extendable to multiclass problems!

Technically

- Optimization of performance function for given predictions depending on threshold t

→ black-box optimization problem of the performance function

Generalized Simulated Annealing

GenSA – for multiclass Threshold Tuning

- The algorithm was developed to simulate the annealing process of molten metal, whose temperature is reduced until it reaches its crystalline state.
- At this state, the metal reaches its minimal thermodynamic energy level, which is a global minimum.
- The metal's thermal energy function is regarded as the objective function.
- During the cooling process, one or more artificial temperatures, which serve as stochastic thermal noise to escape local minima, are added and gradually annealed.
- Depends on two parameters:
  - visiting distribution parameter (higher values make cooling faster)
  - acceptance parameter (higher negative values increase refusal of uphill jumps)

Factor Encoding Benchmark

Settings

- 420 Experiments = Datasets × Impact encoding configurations
- 12 Datasets
  - 3 regression, 7 binary and 2 multiclass classification datasets
  - Each contains at least 1 feature with no less than 10 unique factor levels
- 35 different encoding configurations
  - Integer encoding
  - Dummy encoding
  - Simple: $\lambda = 1$, else $\lambda = 0.5$
  - impact.boundary (ib):

    #{factor levels} ≥ ib ⇒ impact

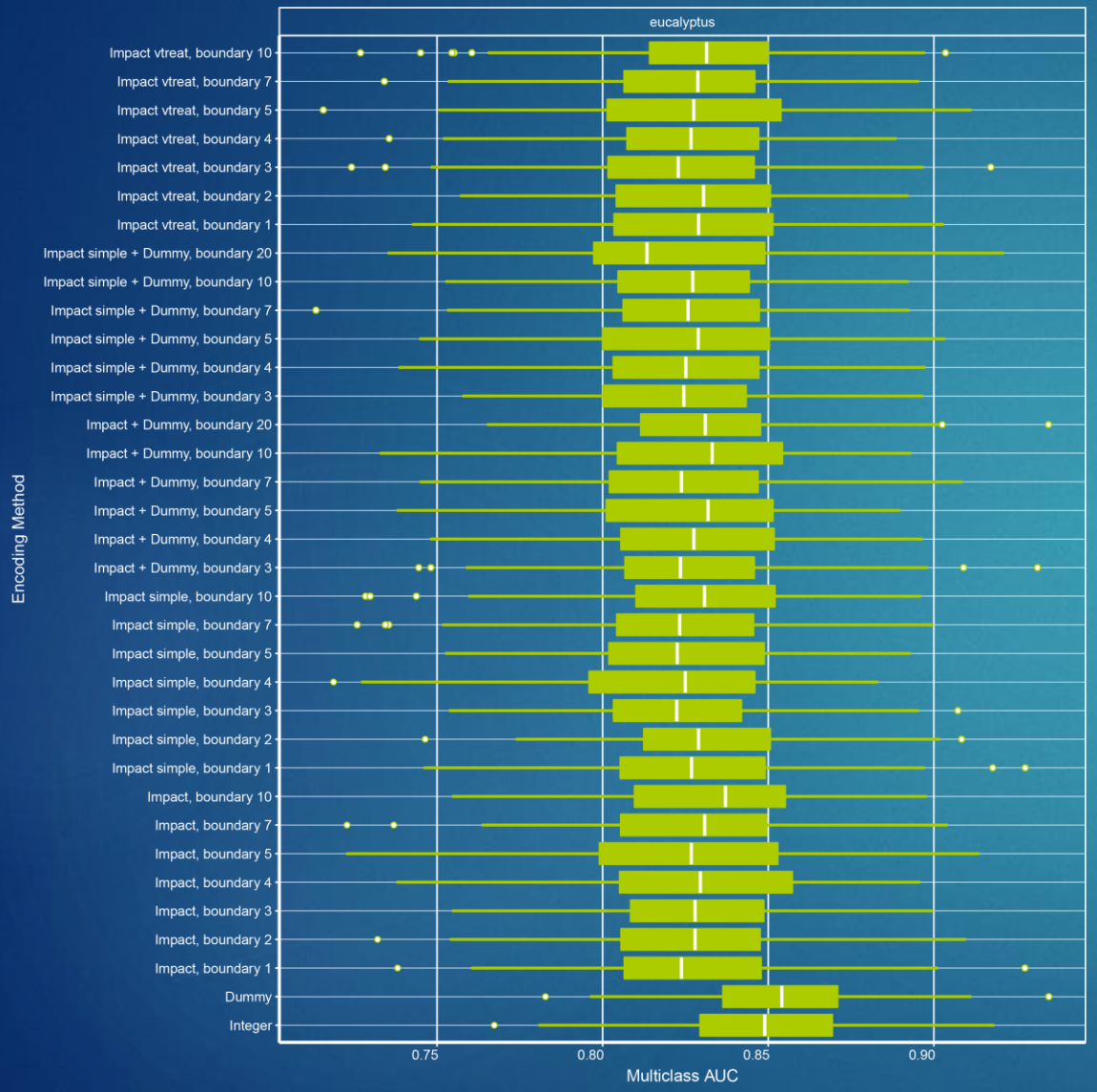
    #{factor levels} < ib ⇒ dummy
  - dummy.boundary (db):

    #{factor levels} ≥ db ⇒ impact

    #{factor levels} < db ⇒ impact + dummy
- XGBoost learner
- 10 × 10 repeated CV

Encoding configurations			
Encoding	simple	impact.boundaries	dummy.boundary
integer dummy impact vtreat	yes/no	1, 2, 3, 4, 5, 7, 10 1, 2, 3, 4, 5, 7, 10	3, 4, 5, 7, 10, 20

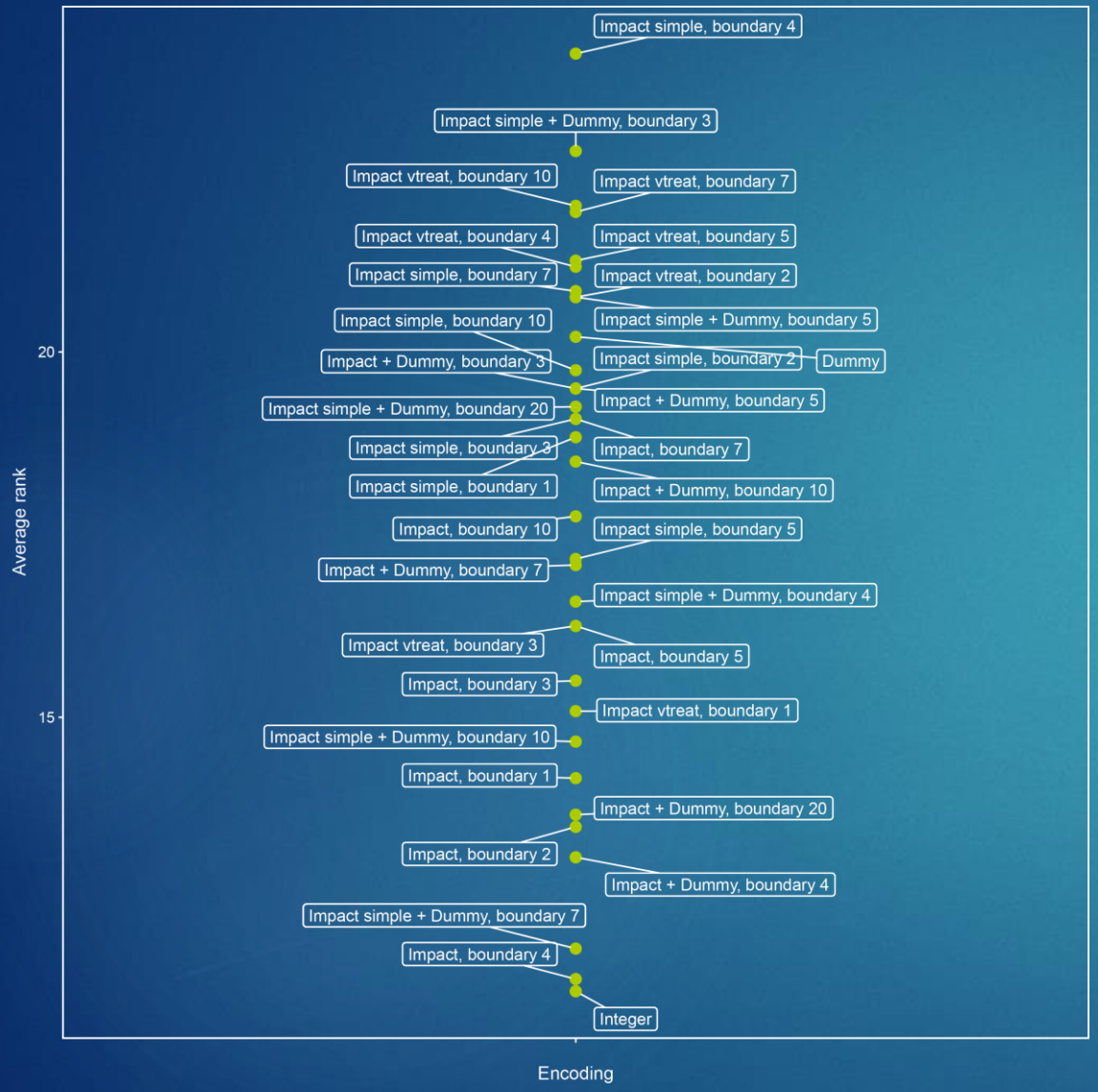
Factor Encoding Benchmark



Results 1/2

- No clear picture on most datasets
- Suprisingly strong performance of integer encoding (and dummy encoding) on some datasets!
- Different results for different performance measures, e.g. mmce, ber, multiclass AUC

Factor Encoding Benchmark



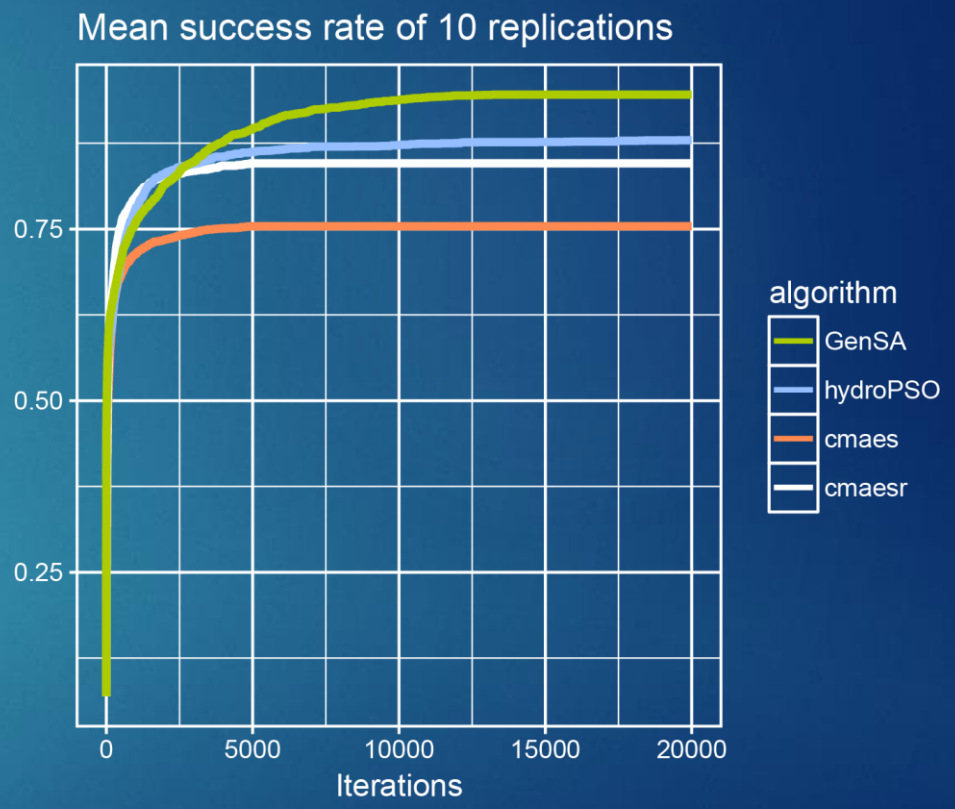
Results 2/2

- Dummy encoding failed on some larger datesets with high cardinality features
- Integer encoding with best average rank for multiclass AUC measure (weighted average 1 vs. 1 multiclass AUC)

Threshold Tuning Benchmark 1

Goal: Find the best optimization algorithm

- 6400 Experiments = Datasets × Optimizers × Learners × Measures × Replications (10)
- 10 multiclass classification datasets
  - up to 10 class levels
  - not too large (< 4000 Observations, < 75 Features)
- 4 Optimization algorithms with starting point $\left(\frac{1}{k}, \ldots, \frac{1}{k}\right)^T$
  - cmaes (mlr's previous standard), evolution strategy approach
  - cmaesr, also allows restarts
  - hydroPSO, particle swarm optimizer
  - GenSA
- Learners: rpart, randomForest, ksvm, naiveBayes
- Measures: mmce, ber, kappa, wkappa



Threshold Tuning Benchmark 2

Goal: Find the best hyperparameters for GenSA - Settings

- 48 000 Experiments = 10 Datasets × 5 Learners × Measures (mmce, ber) × 480 Parameter Configurations

GenSA Parameterset		
Name	values	Description
smooth	TRUE/FALSE	TRUE when the objective function is smooth, or differentiable almost everywhere in the region of par, FALSE otherwise.
temperature	250, 1000, 5000, 10000	Initial value for temperature.
visiting.param	2.1, 2.3, 2.5, 2.7, 2.9	Parameter for visiting distribution.
acceptance.param	0, – 3, – 6, – 9, – 12, – 15	Parameter for acceptance distribution.
simple.function	TRUE/FALSE	FALSE means that the objective function is complicated with many local minima.

Threshold Tuning Benchmark 2

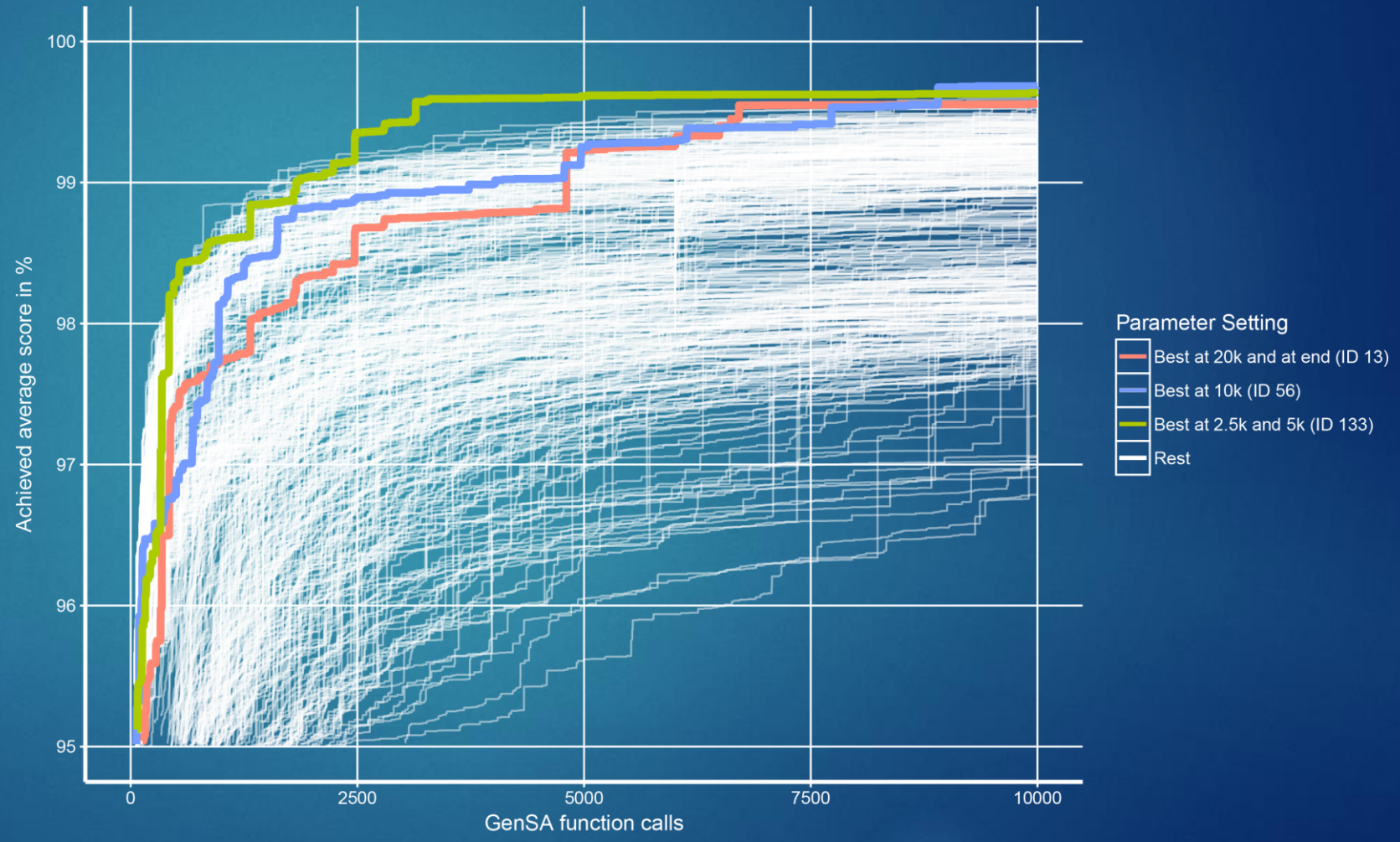
Goal: Find the best hyperparameters for GenSA - Results

Best configuration

- best trade-off between algorithm performance and speed
- ID 133 was chosen

ID 133

- smooth = FALSE
- simple = TRUE
- temperature = 250
- Visit param = 2.5
- accept param = -15



autoxgboost Benchmark

Settings

- Comparison with Auto-WEKA and auto-sklearn
- Same settings as used in the auto-sklearn and auto-WEKA paper
  - same datasets, including train- and test-splits
  - same performance evaluation

- autoxgboost settings:
  - 150 mlrMBO iterations or 10h
  - Integer encoding
  - Threshold tuning (GenSA)
- Additional comparison with featureless, rpart and randomForest

autoxgboost Benchmark

Results

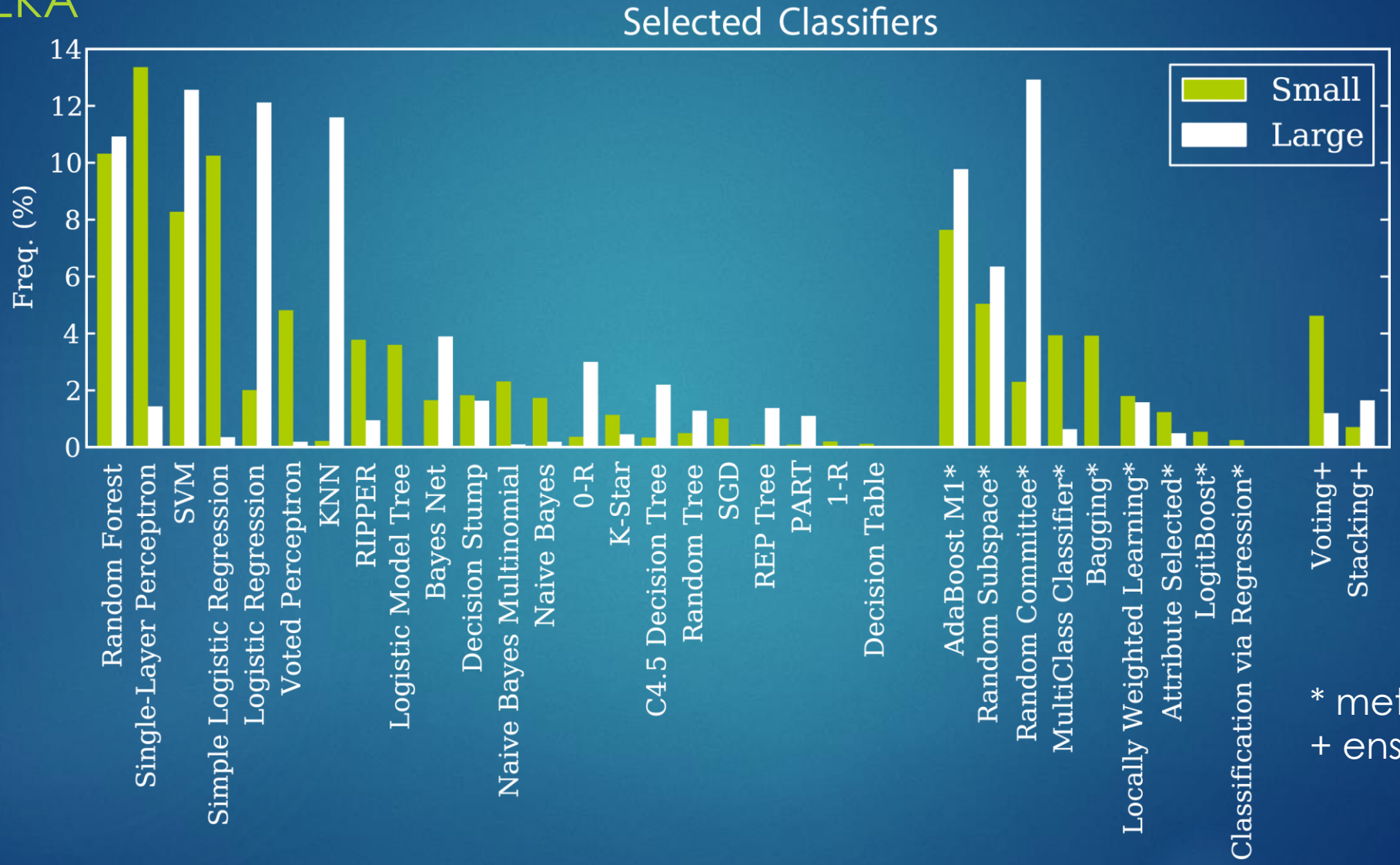
Autoxgboost benchmark results

Dataset	baseline	rpart	randomForest	autoxgboost	Auto-WEKA	auto-sklearn
Dexter	52,78	Error	Error	12.22	7.22	5.56
GermanCredit	32.67	29.67	26.33	27.67	28.33	27.00
Dorothea	6.09	Error	Error	5.22	6.38	5.51
Yeast	68.99	42.25	37.30	38.88	40.45	40.67
Amazon	99.33	74.22	22.00	26.22	37.56	16.00
Secom	7.87	10.43	8.30	7.87	7.87	7.87
Semeion	92.45	36.06	6.92	8.38	5.03	5.24
Car	29,15	5.98	1.54	1.16	0.58	0.39
Madelon	50.26	21.41	25.64	16.54	21.15	12.44
KR-vs-KP	48.96	2.92	1.77	1.67	0.31	0.42
Abalone	84.04	75.98	75.26	73.75	73.02	73.50
Wine Quality	55.68	48.40	32.68	33.70	33.70	33.76
Waveform	68.80	28.33	15.73	15.40	14.40	14.93
Gisette	50.71	7.33	2.57	2.48	2.24	1.62
Convex	50.00	48.80	23.43	22.74	22.05	17.53
Rot. MNIST + BI	88.88	78.19	53.43	47.09	55.84	46.92

Benchmark results are median percent error across 100 000 bootstrap samples (out of 25 runs) simulating 4 parallel runs. Bold numbers indicate best performing algorithms. Best Auto-ML results are represented by green numbers.

Detailed View 1/2

Auto-WEKA

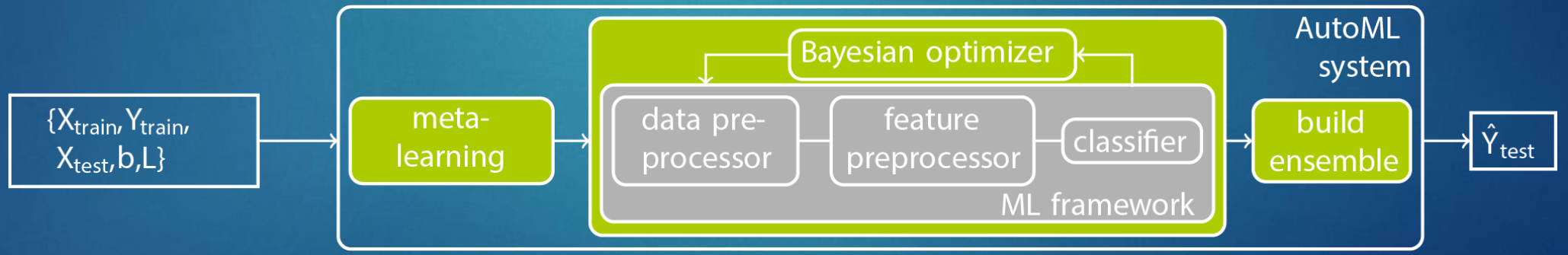


* meta-method
\+ ensembe

Detailed View 2/2

auto-sklearn

- Creates ensembles of size 50 out of the 50 best models via ensemble selection
- When outperformed by Auto-WEKA, in more than 50% of its runs the best classifier it chose is not implemented in scikit-learn (trees with a pruning component)



Conclusion

- autoxgboost simplifies CASH to single learner optimization
- Can compete with with other AutoML systems on various datasets
- Factor Encoding should be worth further research
  - benchmark with more and *better* datasets
  - including more complex feature hashing
- GenSA works quite well for treshold tuning
  - Improved performance and speed!
- Further improvements are possible
  - System switch?
  - Improved factor ecoding
  - More feature engineering (mlrCPO)
  - GPU support

Backup – AutoML Systems

H2O AutoML

- Automatic training of several models and model ensembles
- API for R and python

AutoCompete

- Mainly for participating machine learning competitions
- Serves as starting easy startingpoint to get first predictive models

Backup – AutoML Systems

Tree-based Pipeline Optimization Tool (TPOT)

- "Your Data Science Assistant"
- Uses scikit-learn library
- Uses genetic programming for tuning
- Python based

Main advatange

→ generates standalone scikit-learn python code of the optimal model

Backup – AutoML Systems

pennAI

- From developers of TPOT
- Provides user friendly GUI
- Specialized for complex data in the biomedical and health care domains
- Contains six different ML-learners of scikit-learn
  - Tree
  - knn
  - svm
  - rf
  - gbm
  - Logistic regression (classification) / elastic net (regression)
- Tuning by genetic programming

Backup – Single Learner Systems

Parameter-free STOchastic Learning (PiSTOL)

- Achieves model selection while training
- No need for
  - Parameter tuning or
  - Cross-validation
- Based on stochastic gradient descent
  - Changes step sizes data dependent over time

Main advantage

→ trains up to 7 times faster due to no need for cross-validation

Backup – Single Learner Systems

Driverless AI

- By H2O development team
- Focus on automatic deep learning for entreprises
- Provides automatic feature engeneering and hyperparameter tuning
- Includes tools for model interpretations
  - K-LIME
  - Variable importance
  - Partial dependency plots

Main advatange

→ generates Jupyter notebook containing the python code

Backup – Single Learner Systems

Robust Bayesian Optimization framework (RoBO)

- Uses Bayesian optimization for model tuning
- python based
- Can be applied to differant learning algorithms like
  - Gaussian processes
  - Bayesian neural networks
  - Random forests

tuneranger

- Automatically tunes a random forest of the ranger R package
- Uses mlrMBO for tuning