Where do I begin? Tuning support vector machines and boosted trees

Jill Lundell[1]

[1] Department of Mathematics and Statistics
jflundell@gmail.com

1 August 2019

Summary

Introduction

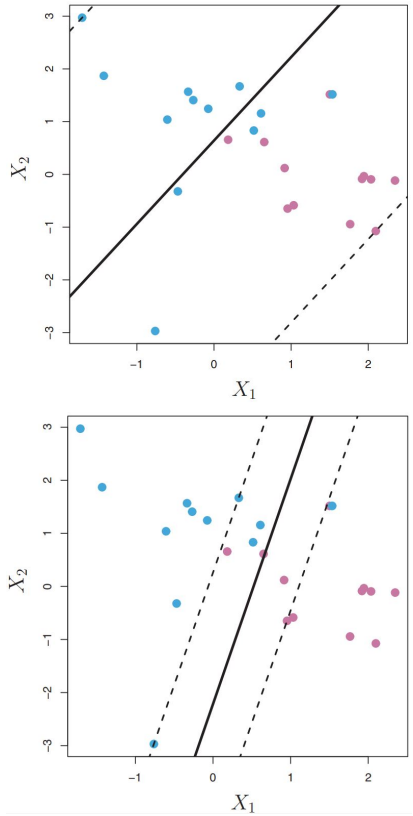
- Define the problem
- Overview of models
- Identify hyperparameter spaces
- Optimization over hyperparameter spaces
- EZtune
- Results

Define the Problem

- Many options for supervised learning models
  - No model is universally superior (no free lunch)
  - Should test several models
- Many models work well if tuned
  - Support vector machines (SVM)
  - Boosted trees (GBMs, adaboost)
- Tuning is not trivial
  - What hyperparameters should be tuned ?
  - What ranges of hyperparameters should be considered ?
  - How do we find a good set of hyperparameters ?

Support vector machines

- Two tuning parameters : cost and γ
- Find hyperplane that separates two classes by maximizing margin between them
- Cost is the tolerance for points being on the wrong side of the margin



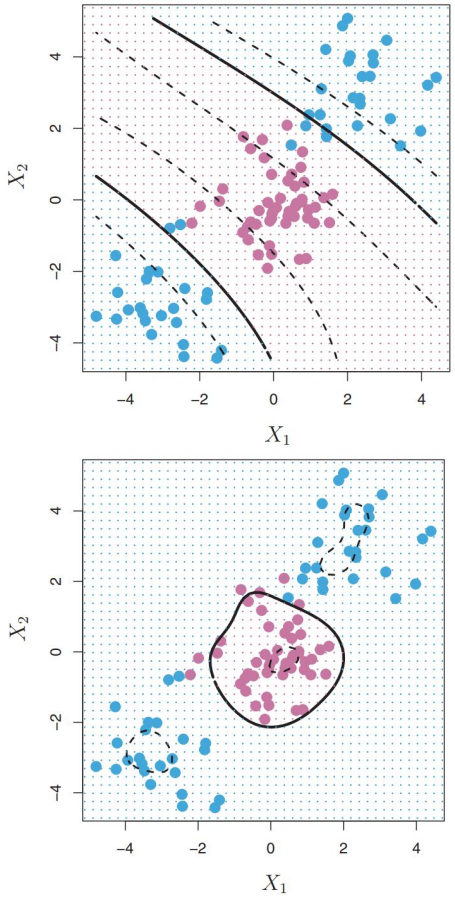
Introduction to statistical learning by James, Witten, Hastie, and Tibshirani

Support vector machines

Model formulation

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i; \gamma)$$

- K is a kernel with tuning parameter γ
- S is the set of support vectors (points on the boundary)
- α_i computed using cost and margin



Introduction to statistical learning by James, Witten, Hastie, and Tibshirani

Boosting

- Idea : Create a strong classifier from many weak classifiers
- Algorithm
  1. Make a small tree from training data
  2. Examine misclassified points
  3. Learn from misclassified points and fit a new tree
  4. Update model by adding new model to old model

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

  5. Go back to step 2 and repeat
- Predictions made by weighted vote of the weak learners

Introduction to statistical learning by James, Witten, Hastie, and Tibshirani

Adaboost

- Learning process
  - Assign increased weights to misclassified points
  - Create new tree by applying weights[2]
  - Add new tree to previous tree
- Tuning parameters
  - Number of iterations
  - Maximum depth of each tree
  - Shrinkage – how fast the tree learns

Gradient boosting machines

- Learning process
  - Create a tree using residuals from previous model
  - Tree constructed by minimizing a loss function using gradient descent[2]
  - Add new tree to the previous tree
- Tuning parameters
  - Number of iterations
  - Maximum depth of each tree
  - Shrinkage – how fast the tree learns
  - Minimum number of observations in terminal nodes

Hyperparameter spaces

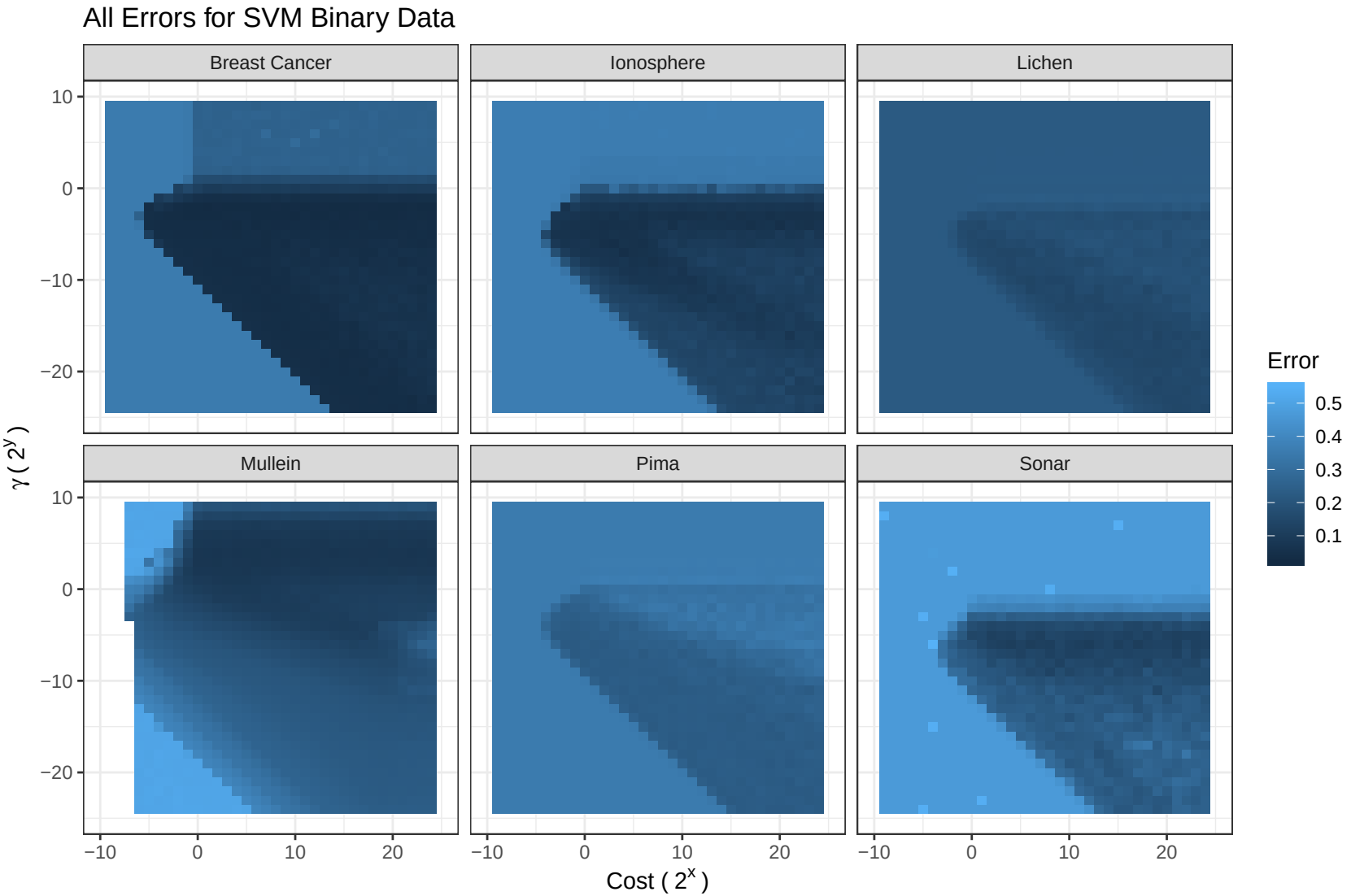
Only restriction is that hyperparameters are greater than 0

Where do we look ?

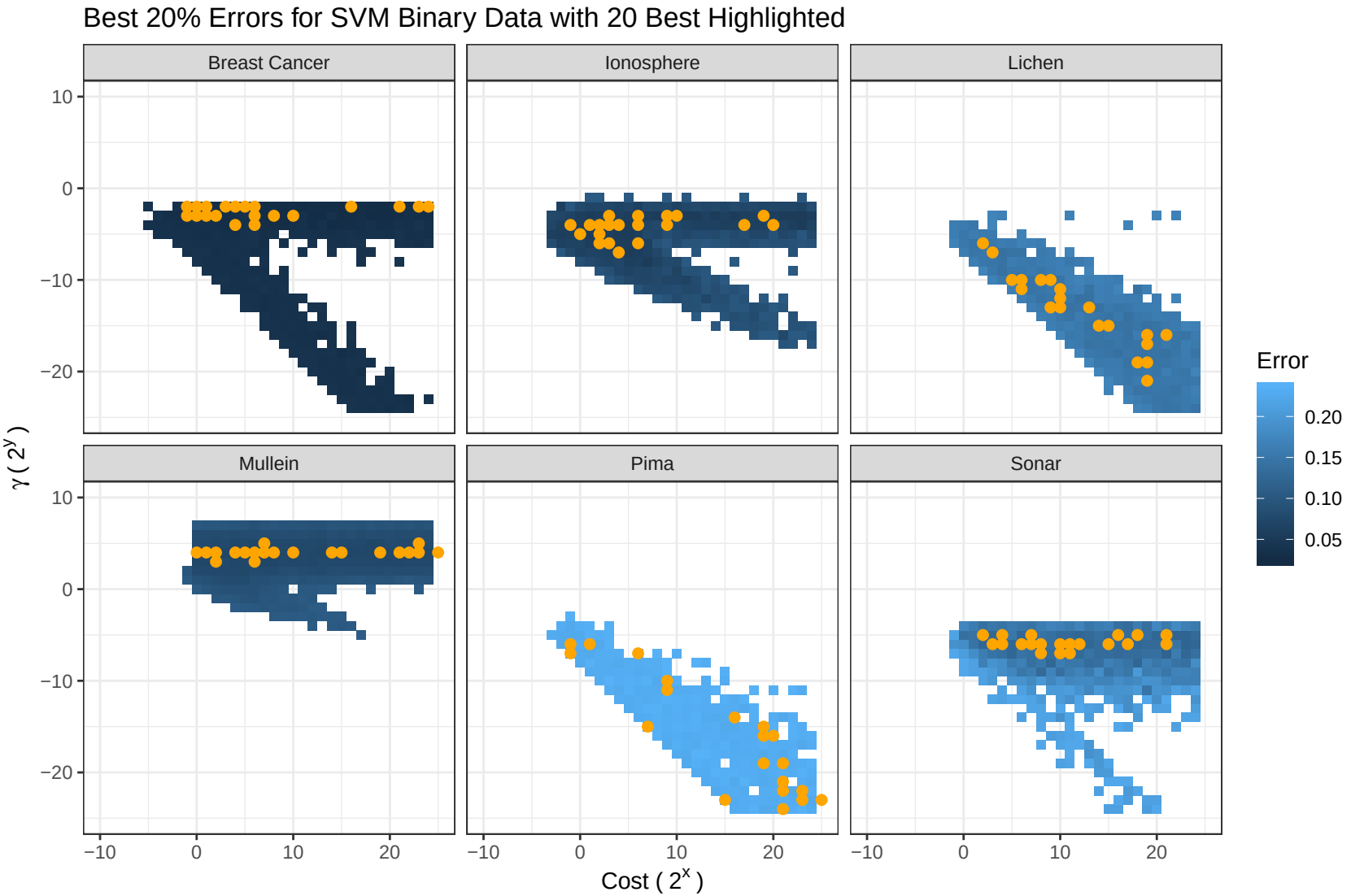
Hyperparameter spaces

- Grid search over an extensive range of hyperparameter values
- Find regions with good accuracies and fast computation times across several datasets
- Binary and regression responses
- Identify good universal starting locations

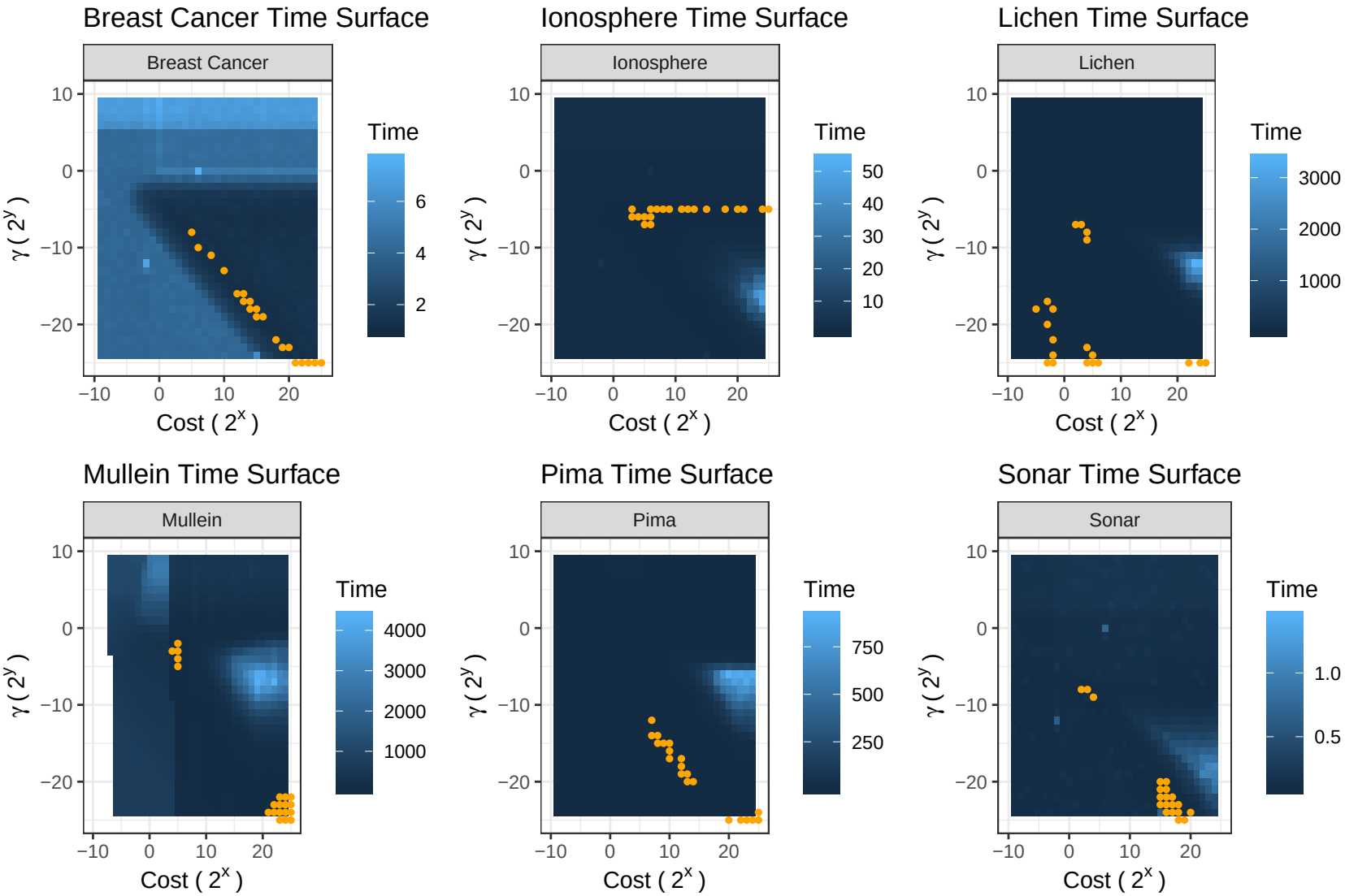
Error surface plots



Best 20% error surface plots



Time surface plots



Parameter spaces for binary classification

Model	Parameter	Ranges	Start
SVM	Cost	[1, 1024]	10
	γ	$[2^{-10}, 2^{10}]$	2^{-5}
GBM	Num trees	[50, 3000]	500
	Tree depth	[1, 15]	5
	Shrinkage	[0.001, 0.1]	0.1
	Min obs	[5, 12]	8
Adaboost	Num trees	[50, 500]	300
	Tree depth	[1, 10]	10
	Shrinkage	[0.01, 0.5]	0.05

Parameter spaces for regression analysis

Model	Parameter	Ranges	Start
SVM	Cost	[1, 1024]	2
	γ	$[2^{-10}, 2^{0}]$	2^{-5}
	ϵ	[0, 0.5]	0.4
GBM	Num trees	[50, 5000]	2000
	Tree depth	[1, 15]	8
	Shrinkage	[0.001, 0.1]	0.1
	Min obs	[5, 10]	5

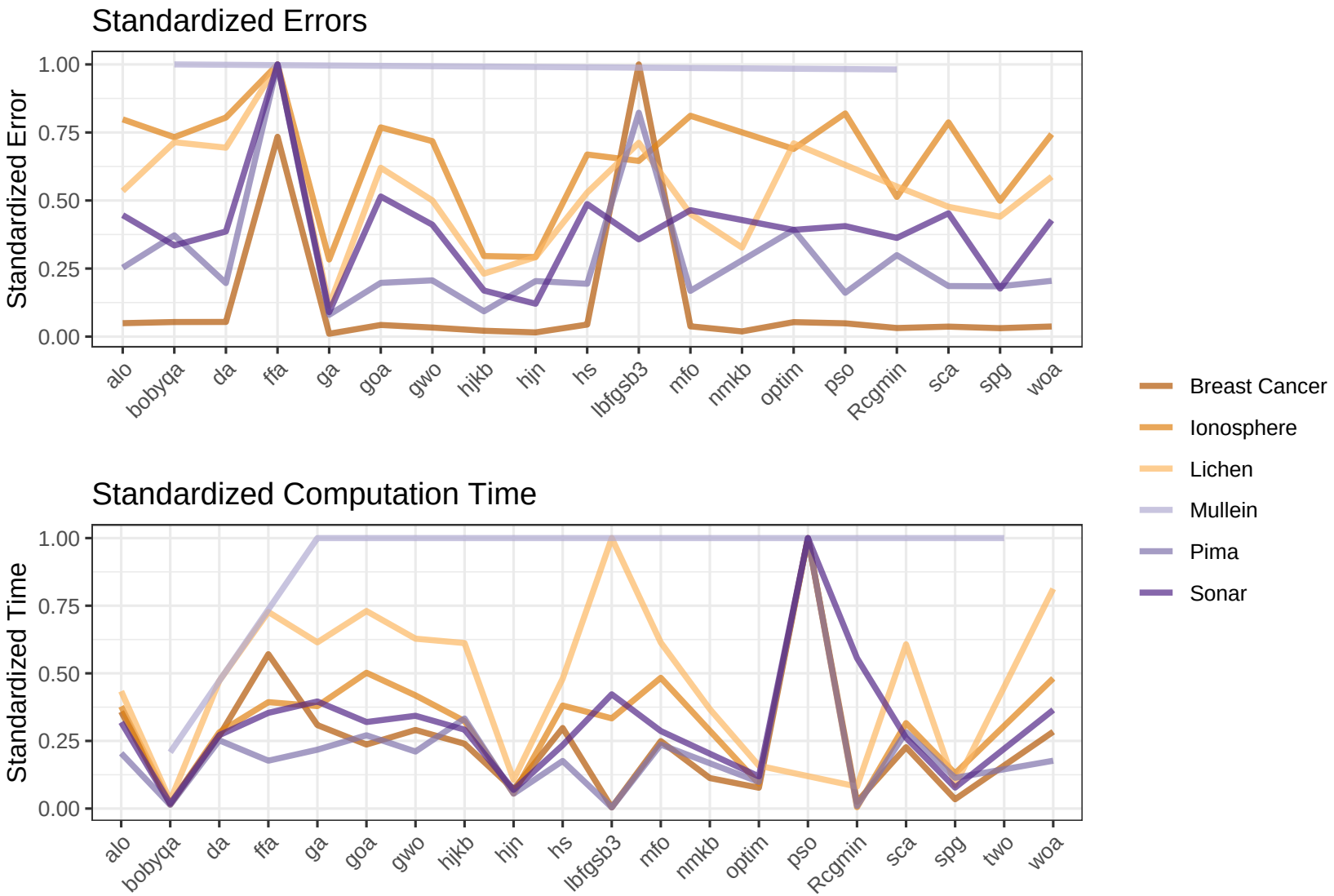
Optimization

How do we look ?

Optimization algorithms

Algorithm	Package	Function
Antlion	MetaheuristicOpt [8]	ALO
BOBYQA	minqa [1]	bobyqa
Dragonfly	MetaheuristicOpt	DA
Firefly	MetaheuristicOpt	FFA
Genetic algorithm	GA [7]	ga
Grasshopper	MetaheuristicOpt	GOA
Grey wolf	MetaheuristicOpt	GWO
Hooke-Jeeves	optimx [4], dfoptim [10]	hjk, hjkb
Improved harmony search	MetaheuristicOpt	HS
L-BFGS	lbfgsb3 [5], stats [6]	lbfgsb3, optim
Moth flame	MetaheuristicOpt	MFO
Nelder-Mead	dfoptim	nmk
Nonlinear conjugate gradient	Rcgmin [3]	Rcgmin
Particle swarm	MetaheuristicOpt	PSO
Sine cosine	MetaheuristicOpt	SCA
Spectral projected gradient	BB [9]	spg
Whale	MetaheuristicOpt	WOA

Optimization analysis - GBM binary classification



Implementation

How do we implement it ?

EZtune

- R package that auto tunes SVMs, GBMs, and adaboost
- Created using hyperparameter space and optimization research
- Produce a good model with reasonable computation time
- Easy to use
- Available on CRAN

eztune options

- Resubstitution, cross validation, or by splitting the data
- Optimization : Hooke-Jeeves or a genetic algorithm
- Model choice : SVM, GBM, or adaboost
- Binary classification or regression
- Fast option
  - Fit on a randomly selected subset of the data
  - Validate using remaining data
  - Can be customized

Package setup

- `eztune` searches over hyperparameter space to find a good model
- `eztune_cv` computes cross validation accuracy or MSE for model

Default arguments :

```
eztune(x, y, method = "svm", optimizer = "hjn",
fast = TRUE, cross = NULL)
```

```
eztune_cv(x, y, model, cross = 10)
```

Results

- `EZtune` model accuracy is often close to best found in grid search
- Does not work well for very small datasets (n < 100)
- Fast option produces good model with short computation time, but larger a training dataset is needed for larger datasets
- Resubstitution does not work well and is slow
- 10-fold cross validation gets best model, but is very slow

The takeaway

- All hyperparameters should be tuned
- Hyperparameter spaces that contain good models across many datasets do exist
- The Hooke-Jeeves and genetic algorithm are able to find good models in these spaces
- Auto tuning using `EZtune` can often find models as accurate about as a large grid search
- Data splitting can find models with error rates that rival 10-fold cross validation and it is much faster
- Never optimize on resubstitution error
- Elastic net tuning will be added to the next version

[1] D. Bates, K. M. Mullen, J. C. Nash, and R. Varadhan. *minqa : Derivative-free optimization algorithms by quadratic approximation*, 2014. R package version 1.2.4.

[2] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA :, 2001.

[3] J. C. Nash. *Rcgmin : Conjugate Gradient Minimization of Nonlinear Functions*, 2014. R package version 2013-2.21.

[4] J. C. Nash et al. On best practice optimization methods in r. *Journal of Statistical Software*, 60(2) :1–14, 2014.

[5] J. C. Nash, C. Zhu, R. Byrd, J. Nocedal, and J. L. Morales. *lbfgsb3 : Limited Memory BFGS Minimizer with Bounds on Parameters*, 2015. R package version 2015-2.13.

[6] R Core Team. *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.

[7] L. Scrucca et al. Ga : a package for genetic algorithms in r. *Journal of Statistical Software*, 53(4) :1–37, 2013.

[8] L. Septem Riza, Iip, and E. Prasetyo Nugroho. *metaheuristicOpt : Metaheuristic for Optimization*, 2017. R package version 1.0.0.

[9] R. Varadhan and P. Gilbert. BB : An R package for solving a large system of nonlinear equations and for optimizing a high-dimensional nonlinear objective function. *Journal of Statistical Software*, 32(4) :1–26, 2009.

[10] R. Varadhan, J. H. University, H. W. Borchers, and A. C. Research. *dfoptim : Derivative-Free Optimization*, 2018. R package version 2018.2-1.