Introduction to Data Science

MODEL SELECTION

BRIAN D'ALESSANDRO

Fine Print: these slides are, and always will be a work in progress. The material presented herein is original, inspired, or borrowed from others' work. Where possible, attribution and acknowledgement will be made to content's original source. Do not distribute, except for as needed as a pedagogical tool in the subject of Data Science.

GOAL OF MODEL SELECTION

The goal of evaluation in model building can be put quite simply:
achieve the ***best generalization performance while avoiding overfitting***.

Remember, in ERM we seek a function f(X) that minimizes the training error (or risk) on our training data.

$$R_{train} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(f(x_i^{train}), y_i^{train})$$

WE ALWAYS want to measure the error on a holdout, or test set, too.

$$R_{test} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(f(x_i^{test}), y_i^{test})$$

We can only be certain our models generalize if we do a correct holdout evaluation. Theory helps us design good algorithms with strong generalization and convergence results, but empirical hold-out testing is always necessary.

Copyright: Brian d'Alessandro, all rights reserved

REMEMBER

Expected Risk != Empirical Risk

Training error is our empirical risk and test set error is our best approximation of expected risk.

Copyright: Brian d'Alessandro, all rights reserved

MODEL SELECTION

When we model, we have many choices. We use holdout evaluation methodologies to optimize the following:

- ***Algorithm Selection***
  (i.e., RF vs. Decision Tree vs. Logistic Regression vs. SVM)

- ***Feature Selection***

- ***Hyper-parameter Selection***, i.e.,
  - "k" in k-NN
  - "C" in SVM
  - MaxDepth, MinLeafSize in Decision Trees
  - Regularization Strength

The optimal configuration of these elements depends on the goals of the problem, which define an appropriate evaluation metric.

Copyright: Brian d'Alessandro, all rights reserved

QUICK HYPER-PARAMETER REVIEW

So far we have learned the following algorithms and their associated complexity parameters.

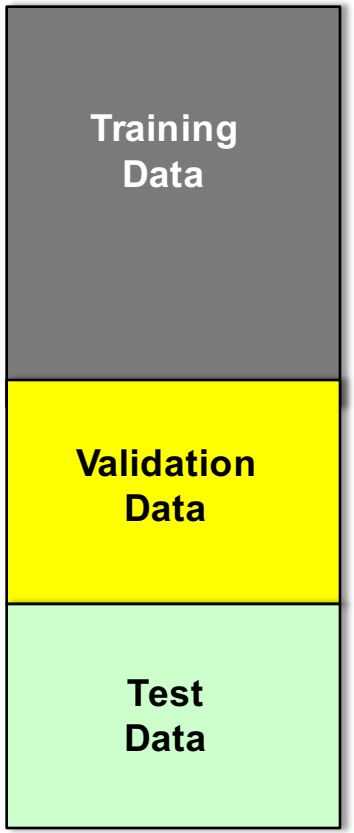
- ***Logistic Regression***
  - C (regularization weight)
  - L1/L2 (regularization strategy)

- ***Support Vector Machine***
  - C (regularization weight)
  - Kernel (and its associated hyperparameters)

- ***Decision Trees***
  - MaxDepth
  - MinLeafSize
  - MinSplitSize

Copyright: Brian d'Alessandro, all rights reserved

BASIC DESIGN FOR HOLDOUT EVALUATION



When doing any sort of model selection, one usually Begins by creating 3 splits of the data.

Training: the training data is used to find the optimal function given the model structure (i.e., fixed algorithm, feature set, hyper-parameters).

Validation: the validation data is used to evaluate the loss/risk for a given model configuration. The configuration with the best loss/risk is selected as the final model

Test: test data is not used for any parameter or model selection. It is only used as a generalization measure.

Copyright: Brian d'Alessandro, all rights reserved

EXAMPLE MODEL SELECTION ROUTINE

Define: Training Data
Define: Validation Data
Define: Test Data

1. For each configuration c in the set C=[Algorithm x Feature Set x Hyper-parameter]:

- Find the function $\hat{f}_c$ such that: $\hat{f}_c = \underset{f \in \mathbb{F}}{\operatorname{argmin}} R^{train}$
- With $\hat{f}_c$ estimated, get the validation loss: $R_c^{val} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(\hat{f}_c(x_i^{val}), y_i^{val})$

2. Choose the optimal configuration c_{opt} such that: $c_{opt} = \underset{c \in C}{\operatorname{argmin(max)}} R_c^{val}$

3. Define: NewTrain = Train + Validation data

4. Find the function $\hat{f}$ such that: $\hat{f} = \underset{f \in \mathbb{F}}{\operatorname{argmin}} R^{NewTrain}$

5. Estimate the test loss as: $R^{test} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(\hat{f}(x_i^{test}), y_i^{test})$

Copyright: Brian d'Alessandro, all rights reserved

SOME NOTES ON VALIDATION RISK

The validation loss metric does not have to be the same as the training loss.

- Sometimes we need a loss metric for an application that is not very easy or even possible to directly minimize.
- I.e., we use logistic-loss to find *f* but we really need a loss metric that enables optimal ranking (such as AUC)
- We might want to minimize a training loss metric, but maximize the validation metric (again logistic-loss vs. AUC)

- With $\hat{f}_c$ estimated, get the validation loss: $R_c^{val} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{L}(\hat{f}_c(x_i^{val}), y_i^{val})$

Copyright: Brian d'Alessandro, all rights reserved

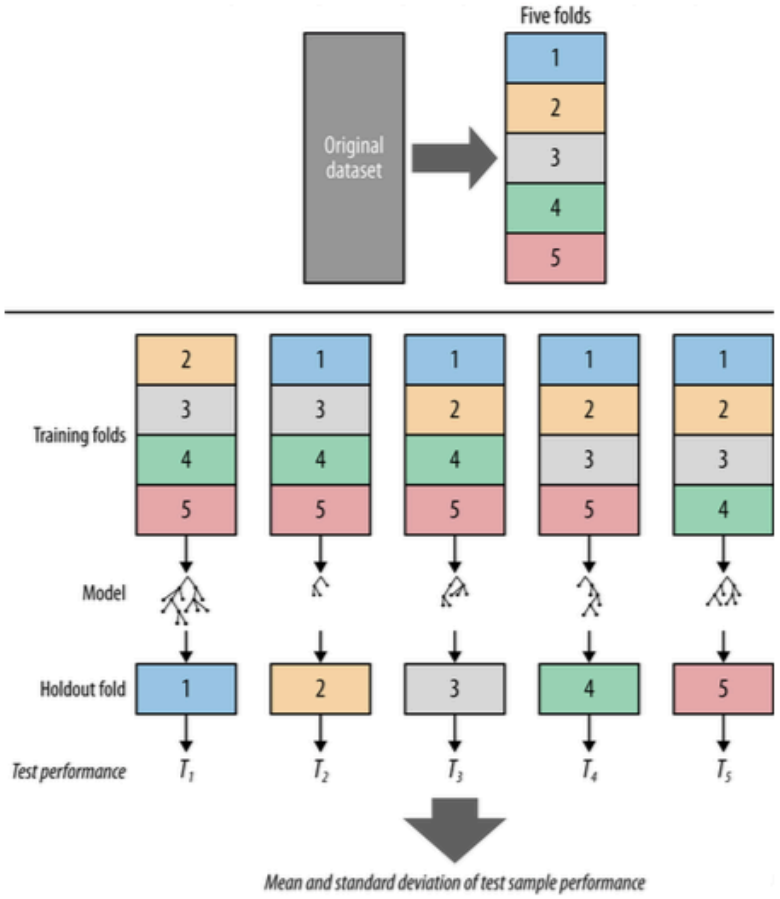
CROSS-VALIDATION

Sometimes we don't have enough data to do a single train/validation/test split.

We can "recycle" data by using k-fold cross validation as our validation scheme.

Copyright: Brian d'Alessandro, all rights reserved

ANOTHER ILLUSTRATION



Important properties of cross-validation:

- Each record contributes to both training and testing
- Use of multiple folds enables us to compute variance statistics and do statistical inference

Copyright: Brian d'Alessandro, all rights reserved

EXAMPLE X-VALIDATION SCHEME

Define: Training Data
Define: Test Data

1. For each configuration c in the set C=[Algorithm x Feature Set x Hyper-parameter]:

- For each fold k:
  - Find the function $\hat{f}_c^k$ such that: $\hat{f}_c^k = \underset{f \in \mathbb{F}}{\operatorname{argmin}} R^{train-k}$
  - With $\hat{f}_c^k$ estimated, get the validation loss: $R_c^{val-k} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(\hat{f}_c(x_i^{val-k}), y_i^{val-k})$
- Get the average loss across all folds: $R_c^{val} = \sum_{i=1}^{k} R_c^{val-k}$

2. Choose the optimal configuration c_{opt} such that: $c_{opt} = \underset{c \in C}{\operatorname{argmin}(\max)} R_c^{val}$

3. Find the function $\hat{f}$ such that: $\hat{f} = \underset{f \in \mathbb{F}}{\operatorname{argmin}} R^{train}$ **using** c_{opt}

4. Estimate the test loss as: $R^{test} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(\hat{f}(x_i^{test}), y_i^{test})$

all rights reserved

SOME HOLDOUT TESTING NOTES

How should we choose split size or k?

There is no golden rule that defines how big training, validation and test sets should be.

You want each data set to be big enough to reduce the variance of your estimates (both training and test).

Estimation variance comes in two flavors – variance of the function being fit to the training data and variance of the estimate of the validation/test risk.

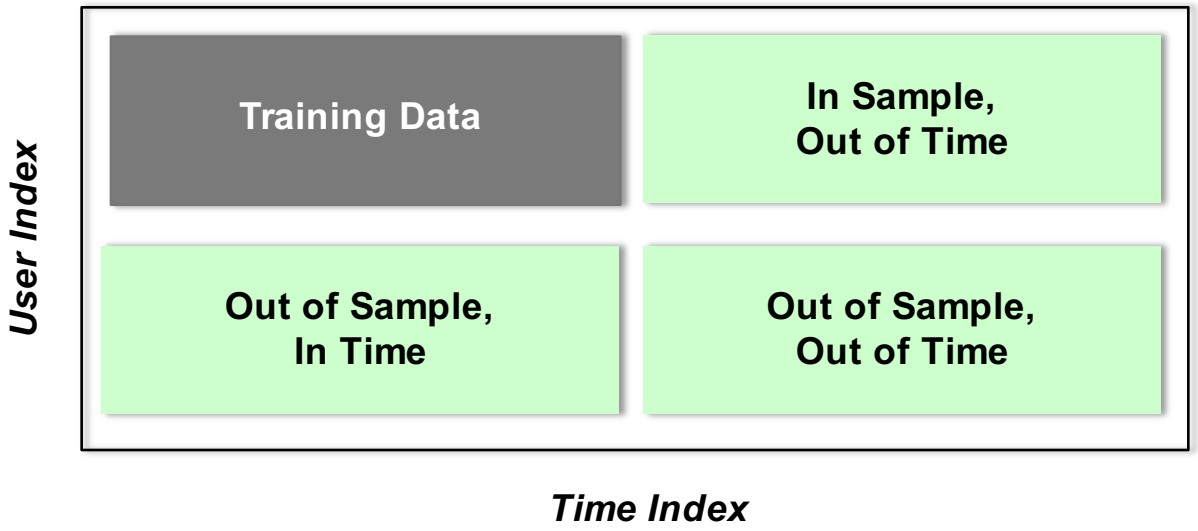
Good heuristics are 70%/20%/10% for train/val/test splits and k=5 or 10 for x-validation.

Also, data should be randomly split with no overlap in instances between sets when there is no time dimension. If dealing with temporal data, consider splitting on time dimensions as well.

Copyright: Brian d'Alessandro, all rights reserved

SPLITTING SCHEMES

Splitting data for a holdout isn't always as straightforward as taking a random sample, but it is also simple if you remember the core rule:
no overlap between training and test!



If there is no time index, out-of-sample is sufficient. With a time index, you would want to sample different users from a different time period (usually future)

Copyright: Brian d'Alessandro, all rights reserved

EXAMPLE MODEL SELECTION

We have a dataset with 14 features, roughly 10k examples and only 253 positive examples.

Goal

Build a classifier that has good ranking properties.

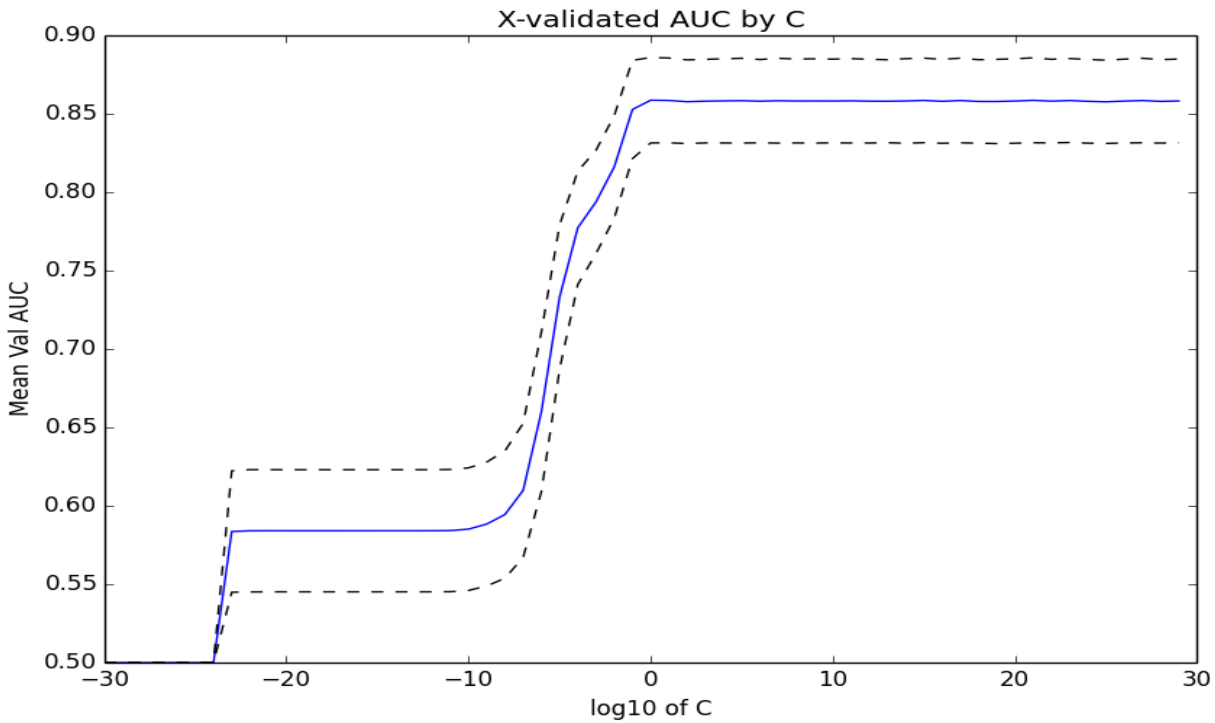
Solution

1. We'll use Logistic Regression because it is robust in small-sample sizes and imbalanced classes, and also returns a score instead of just label predictions
2. With small data we expect high variance, so we need to use regularization
3. We'll split data into 80/20 train/test and run 10-fold x-validation on train.
4. We'll use AUC to choose a regularization weight

Copyright: Brian d'Alessandro, all rights reserved

RESULTS OF XVALIDATION

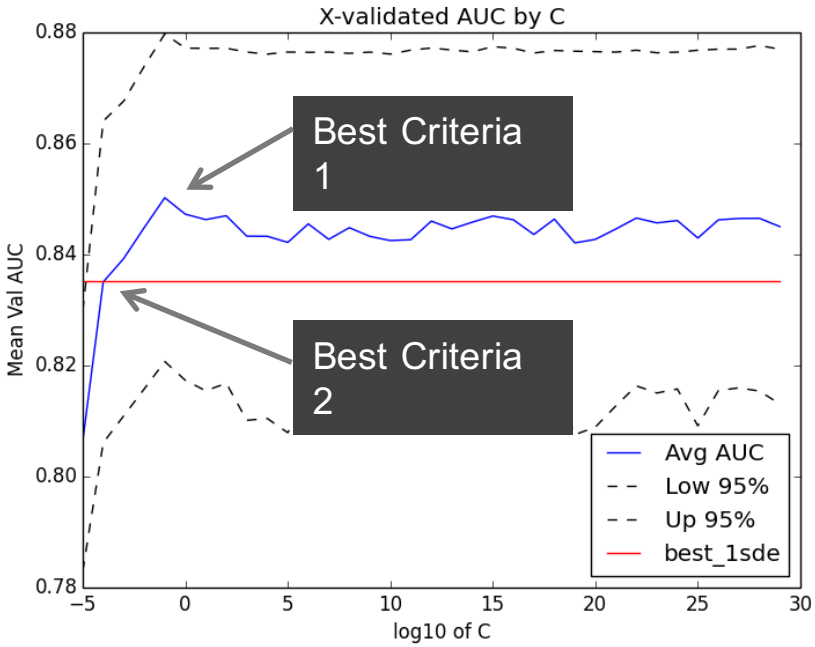
With less regularization we actually do better here (strong signal in the features). But we still see that between 1 to 10^{30} we get nearly the same results. It would be better if we can zoom into the region where performance is better.



Copyright: Brian d'Alessandro, all rights reserved

RULE FOR CHOOSING BEST OPTION

When we zoom in we can see that statistically speaking, everything above C=1 is essentially the same (i.e., strongly overlapping confidence intervals). There are two ways to choose C here.



Criteria 1:
Choose option with max(Xval AUC)

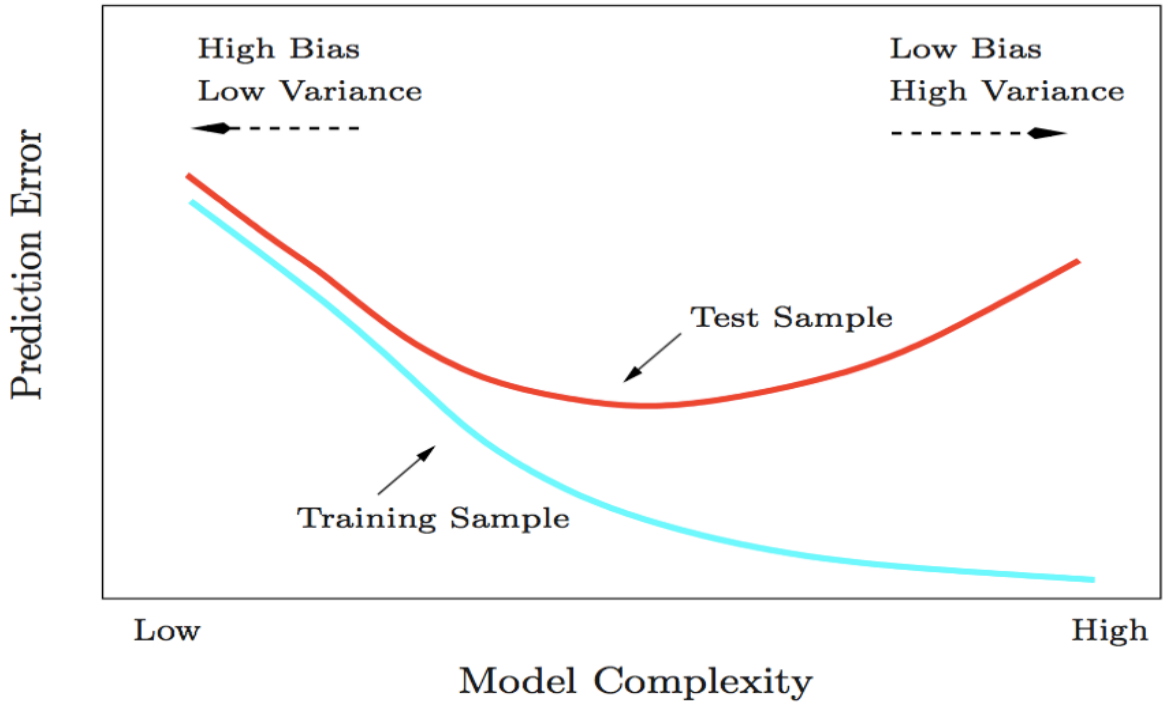
Criteria 2:
Find all options where AUC>=max(AUC) – 1stderror

Choose least complex option (highest regularization)

Copyright: Brian d'Alessandro, all rights reserved

FINDING THE PERFORMANCE SWEET SPOT

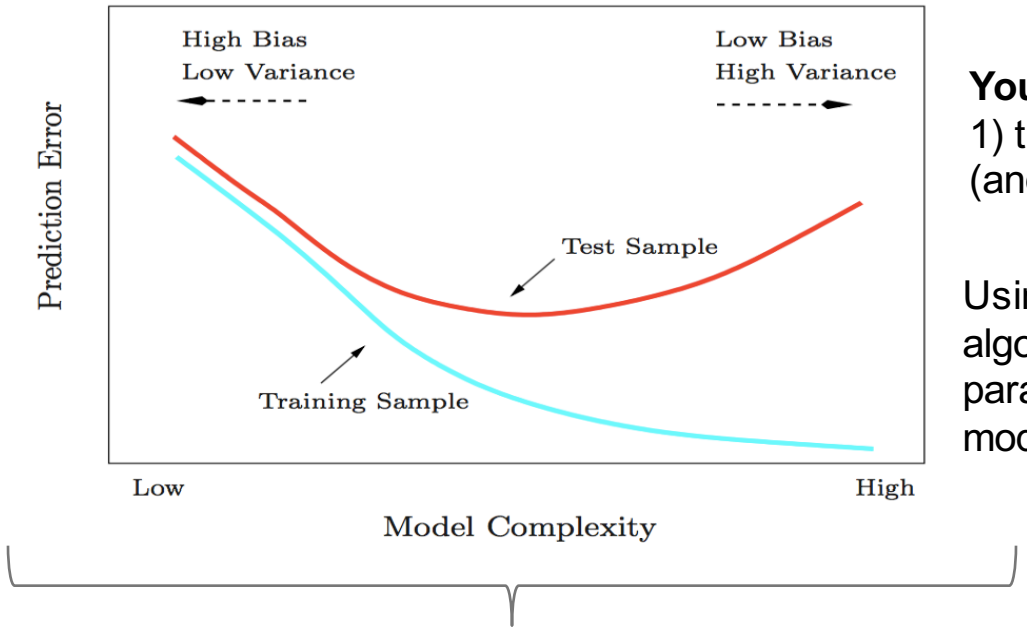
While theory motivates the algorithms we use, finding the optimal bias-variance tradeoff is generally always an empirical process. The routines presented here give us generally our best chance at being successful.



Copyright: Brian d'Alessandro, all rights reserved

ON CHOOSING HYPER-PARAMETERS

Software will do most of the model training work, but the scientist still needs to instruct the software on which hyper-parameters to explore.



Range should start here

And end here

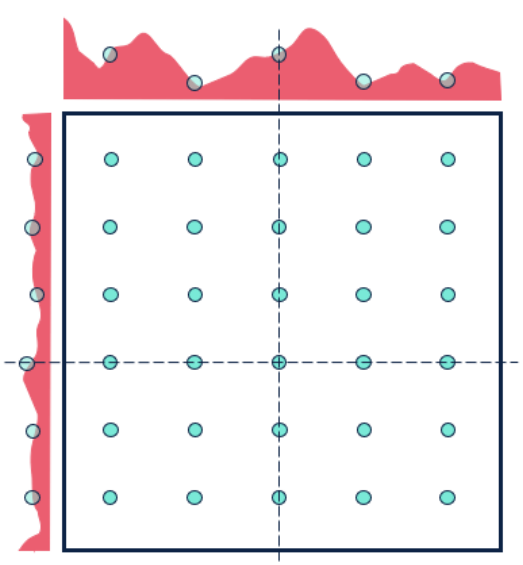
You have 2 main choices to make:
1) the range of values to explore, 2) the actual (and number of) values to explore

Using your best judgment and knowledge of the algorithm and problem, you want hyper-parameters that span the range of low to high model complexity

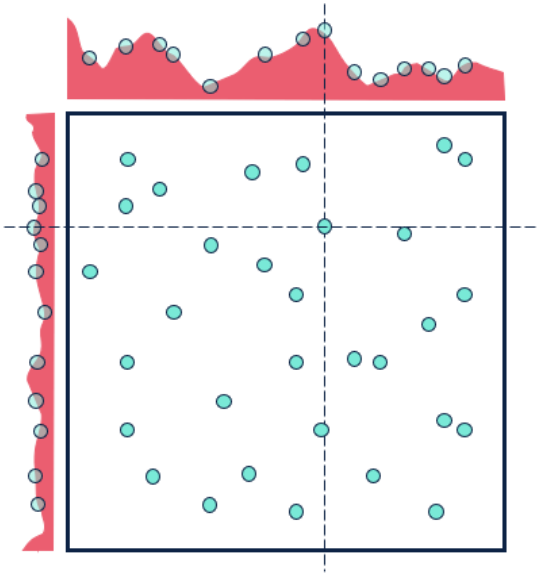
Copyright: Brian d'Alessandro, all rights reserved

ON CHOOSING HYPER-PARAMETERS

When optimizing multiple hyper-parameters, the search starts to become more expensive. One way to optimize learning per unit time is to randomly select values within the selected range.



Grid Search



Random Search

In many cases some hyper-parameters are more important than others for tuning the model.

With grid search, you'll waste learning opportunities by repeatedly training on the same value of a particular hyper-parameter

With random search, each training iteration uses new values of each hyper-parameter, allowing for broader exploration of the space.

Good reference:
https://www.jmlr.org/papers/v13/bergstra12a.html

Copyright: Brian d'Alessandro, all rights reserved