Hyperparameters and Model Selection

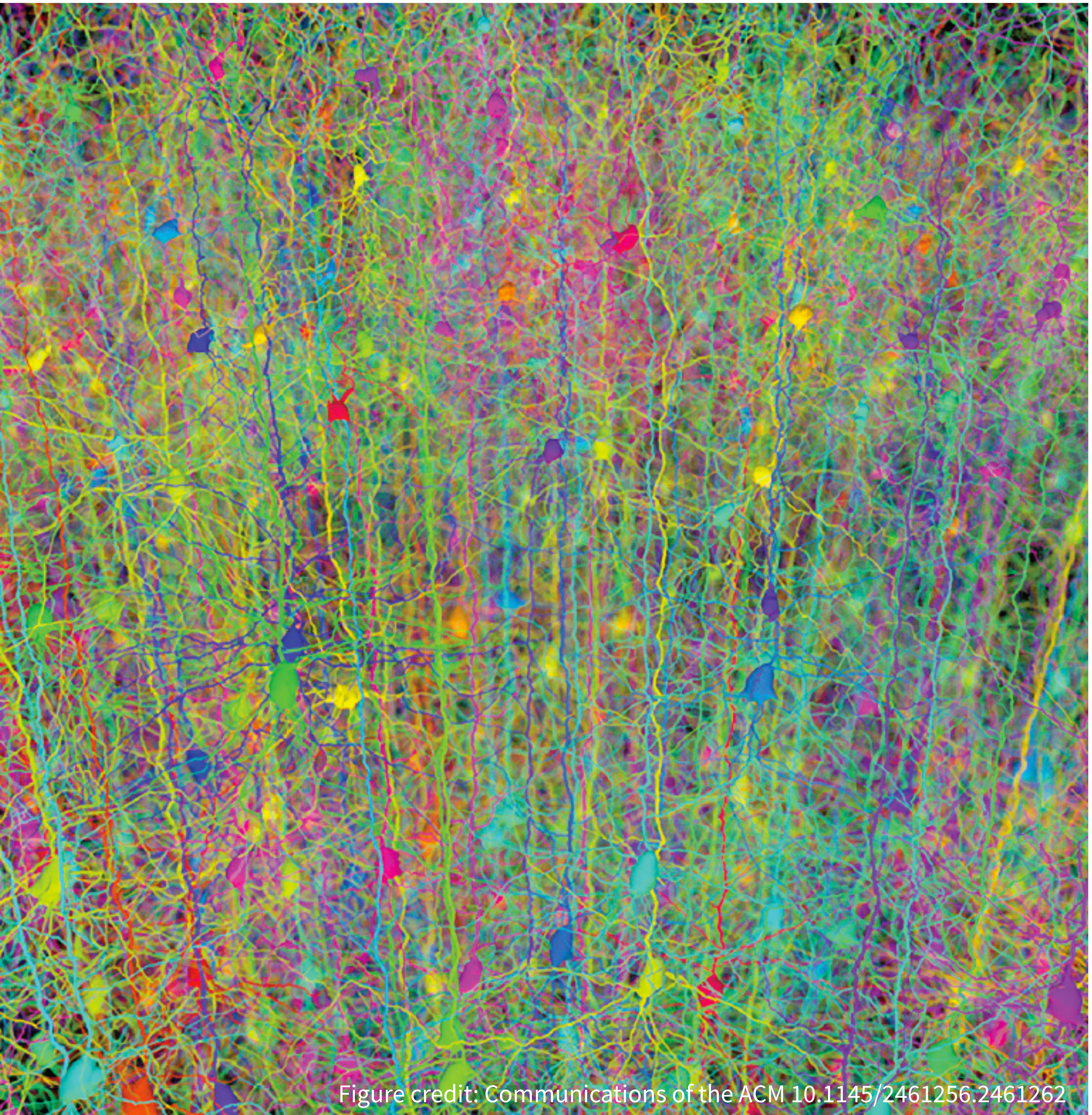


Figure credit: Communications of the ACM 10.1145/2461256.2461262

GRAHAM TAYLOR

VECTOR INSTITUTE

SCHOOL OF ENGINEERING
UNIVERSITY OF GUELPH

CANADIAN INSTITUTE
FOR ADVANCED RESEARCH

UNIVERSITY of GUELPH
CHANGING LIVES IMPROVING LIFE

CIFAR
CANADIAN INSTITUTE FOR ADVANCED RESEARCH

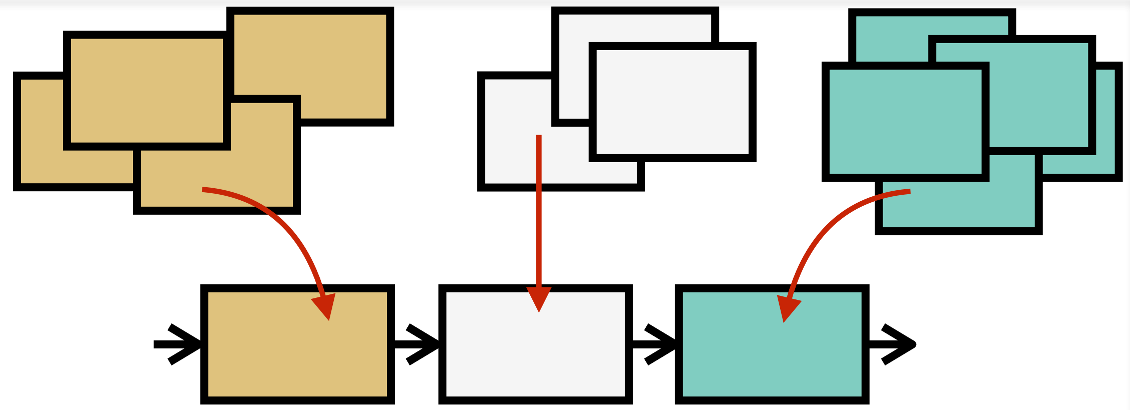
Model vs. Learning Algorithm

Model vs. Learning Algorithm

Model (Architecture)

Describes path from input to output

Employed at test time and training time

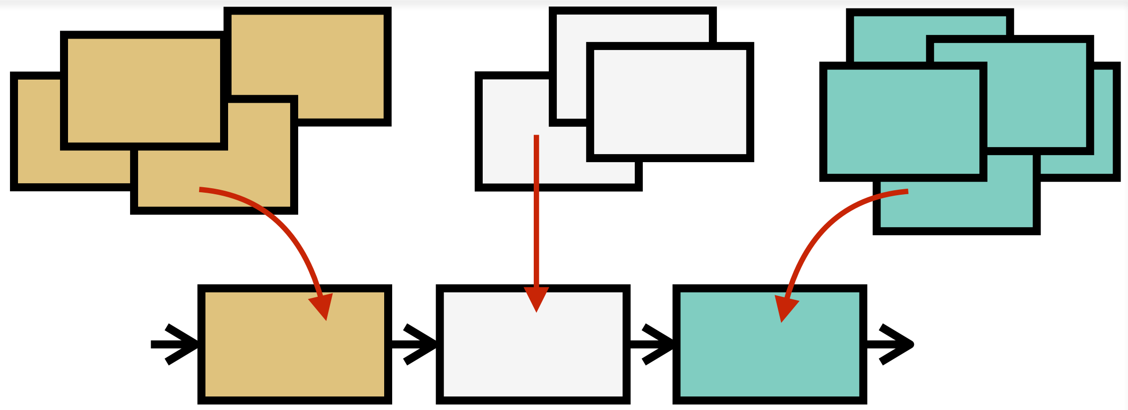


Model vs. Learning Algorithm

Model (Architecture)

Describes path from input to output

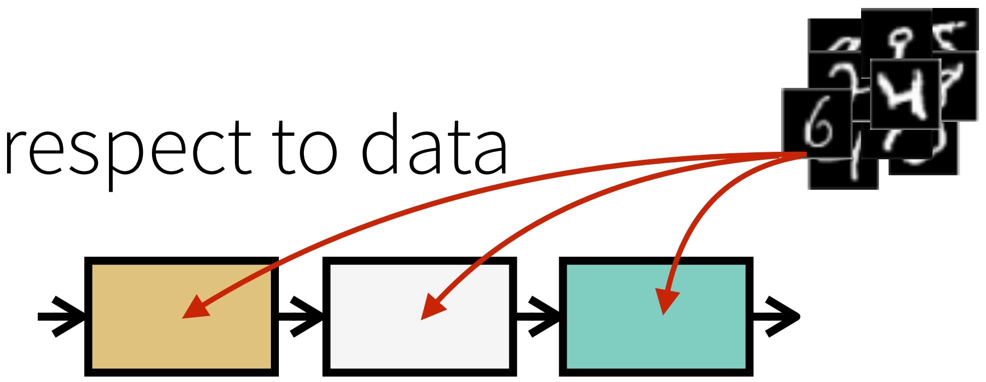
Employed at test time and training time



Learning Algorithm

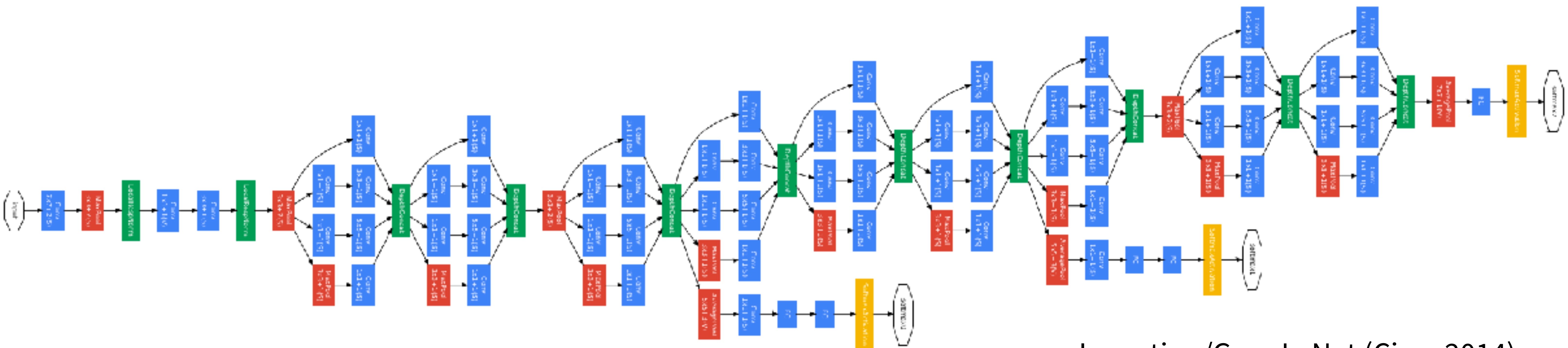
Describes how parameters are updated with respect to data

Employed at training time



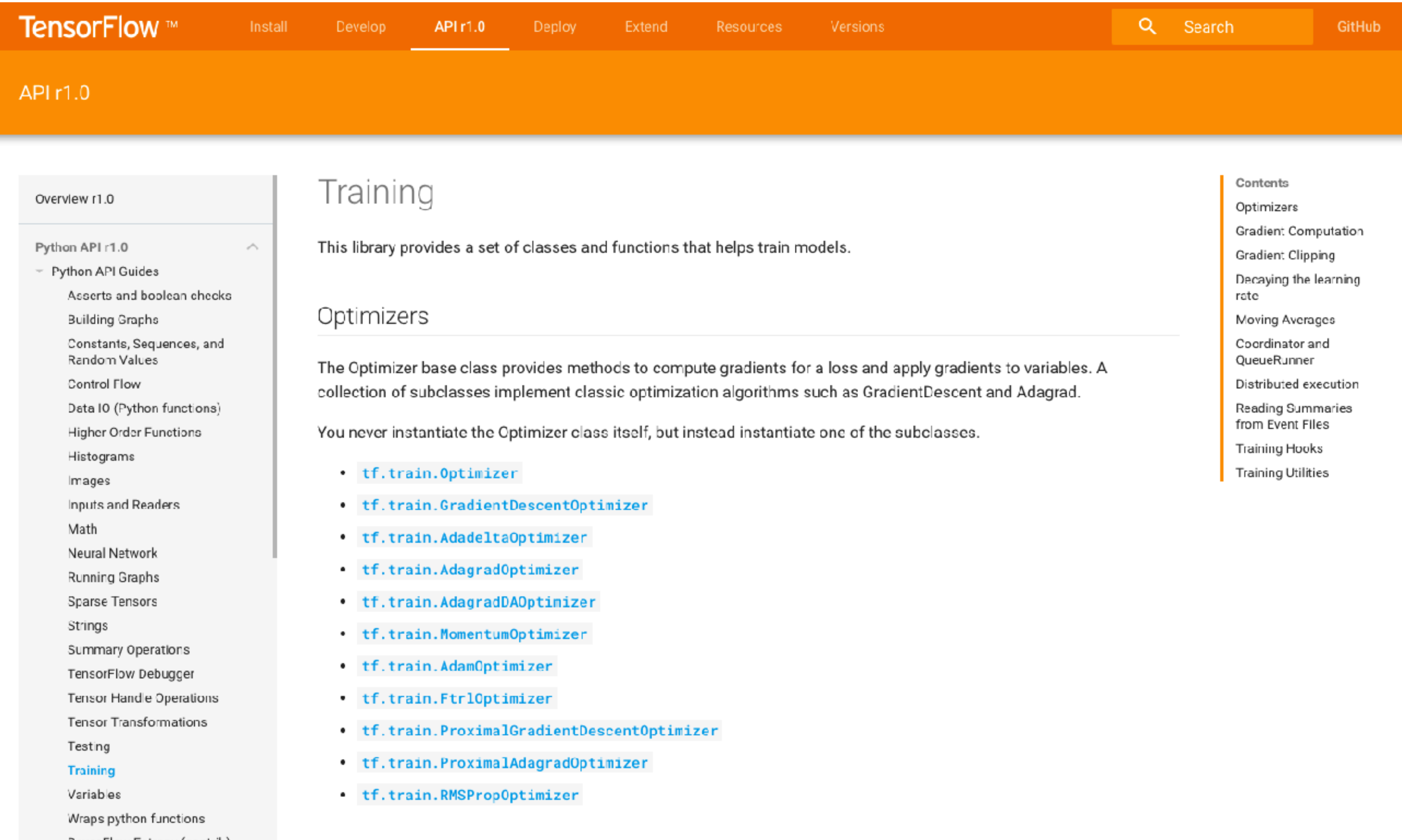
Trend: Complex ML Architectures

	depth	size	top-5-error (%)	improved top-5 error (%)
AlexNet (2012)	8	60M	17.0	15.3
Inception (2014)	22	4M	10.1	6.7
VGGNet (2014)	19	144M	8.0	6.8
ResNet (2015)	152	~45M	5.7	3.6



Inception/GoogLeNet (Circa 2014)

Trend: Flexibility in Choosing a Learning Algorithm



TensorFlow ™ Install Develop API r1.0 Deploy Extend Resources Versions Search GitHub

API r1.0

Overview r1.0

Python API r1.0

Python API Guides

- Asserts and boolean checks
- Building Graphs
- Constants, Sequences, and Random Values
- Control Flow
- Data IO (Python functions)
- Higher Order Functions
- Histograms
- Images
- Inputs and Readers
- Math
- Neural Network
- Running Graphs
- Sparse Tensors
- Strings
- Summary Operations
- TensorFlow Debugger
- Tensor Handle Operations
- Tensor Transformations
- Testing
- Training
- Variables
- Wraps python functions

Training

This library provides a set of classes and functions that helps train models.

Optimizers

The Optimizer base class provides methods to compute gradients for a loss and apply gradients to variables. A collection of subclasses implement classic optimization algorithms such as GradientDescent and Adagrad.

You never instantiate the Optimizer class itself, but instead instantiate one of the subclasses.

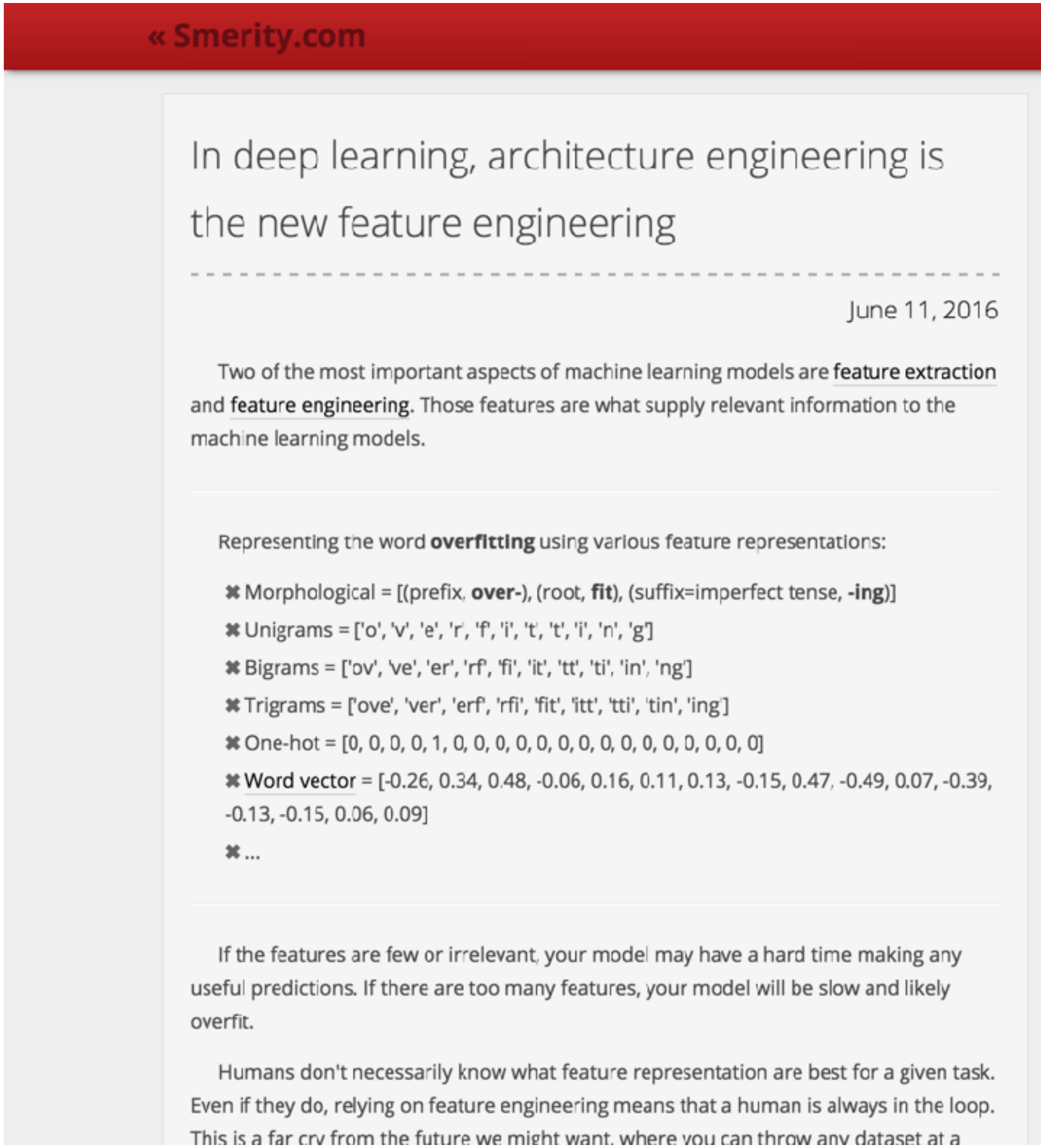
- `tf.train.Optimizer`
- `tf.train.GradientDescentOptimizer`
- `tf.train.AdadeltaOptimizer`
- `tf.train.AdagradOptimizer`
- `tf.train.AdagradDAOptimizer`
- `tf.train.MomentumOptimizer`
- `tf.train.AdamOptimizer`
- `tf.train.FtrlOptimizer`
- `tf.train.ProximalGradientDescentOptimizer`
- `tf.train.ProximalAdagradOptimizer`
- `tf.train.RMSPropOptimizer`

Contents

via commons.wikimedia.org

Engineering architectures

- Romanticized notion of DL - end of feature engineering
- Feature engineering has decreased
- Architectures have become more complex



« Smerity.com

In deep learning, architecture engineering is the new feature engineering

June 11, 2016

Two of the most important aspects of machine learning models are feature extraction and feature engineering. Those features are what supply relevant information to the machine learning models.

Representing the word **overfitting** using various feature representations:

- Morphological = [(prefix, **over-**), (root, **fit**), (suffix=imperfect tense, **-ing**)]
- Unigrams = ['o', 'v', 'e', 'r', 'f', 'i', 't', 't', 'i', 'n', 'g']
- Bigrams = ['ov', 've', 'er', 'rf', 'fi', 'it', 'tt', 'ti', 'in', 'ng']
- Trigrams = ['ove', 'ver', 'erf', 'rfi', 'fit', 'itt', 'tti', 'tin', 'ing']
- One-hot = [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
- Word vector = [-0.26, 0.34, 0.48, -0.06, 0.16, 0.11, 0.13, -0.15, 0.47, -0.49, 0.07, -0.39, -0.13, -0.15, 0.06, 0.09]
- ...

If the features are few or irrelevant, your model may have a hard time making any useful predictions. If there are too many features, your model will be slow and likely overfit.

Humans don't necessarily know what feature representation are best for a given task. Even if they do, relying on feature engineering means that a human is always in the loop. This is a far cry from the future we might want, where you can throw any dataset at a

http://smerity.com/articles/2016/architectures_are_the_new_feature_engineering.html

Model Selection

You've got a task and you've got a dataset.

How do you choose a model, learning algorithm, and all of the associated tunable "knobs"?

Example: Scikit-learn

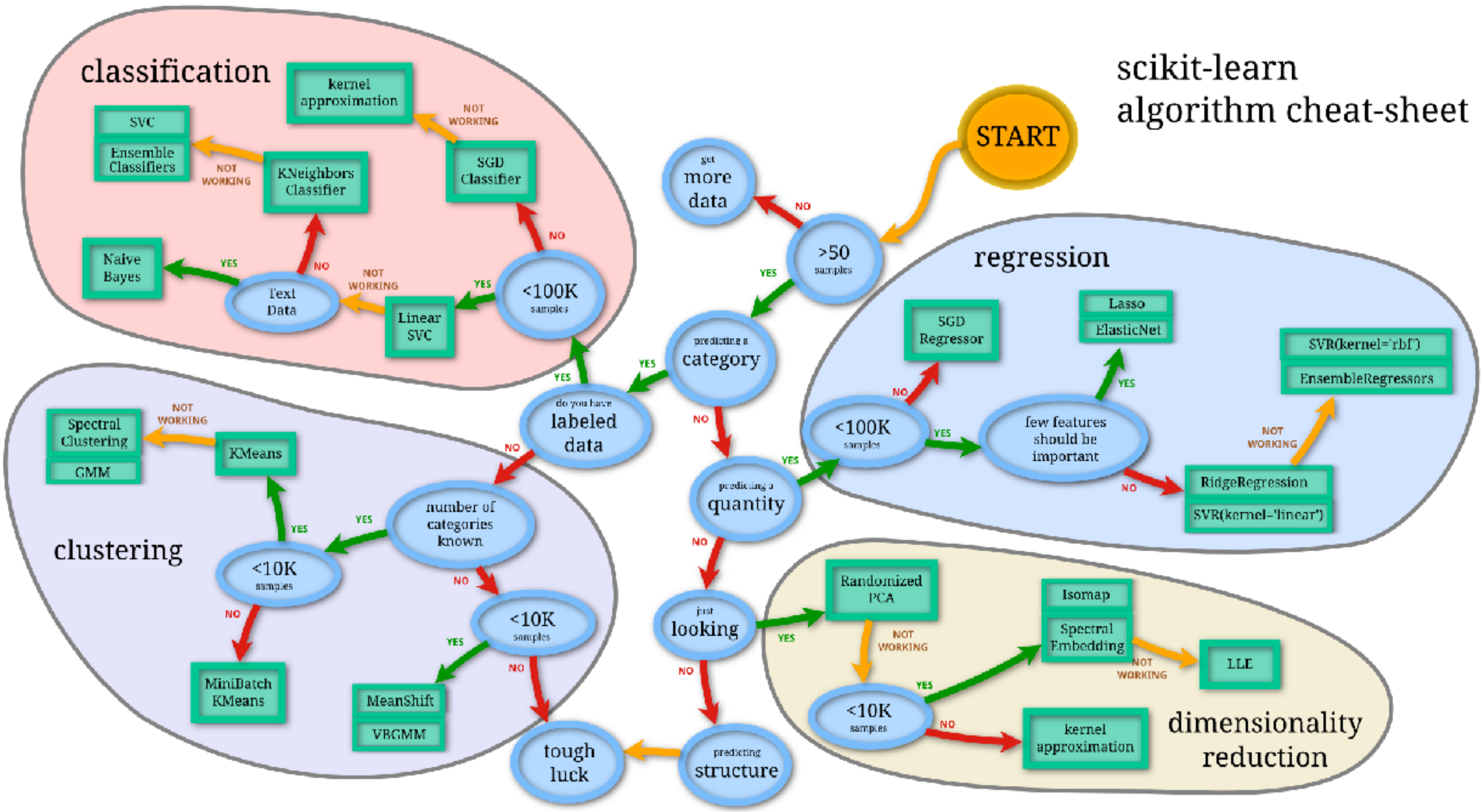


Image Credit: Scikit-learn (http://scikit-learn.org)

Hyperparameters

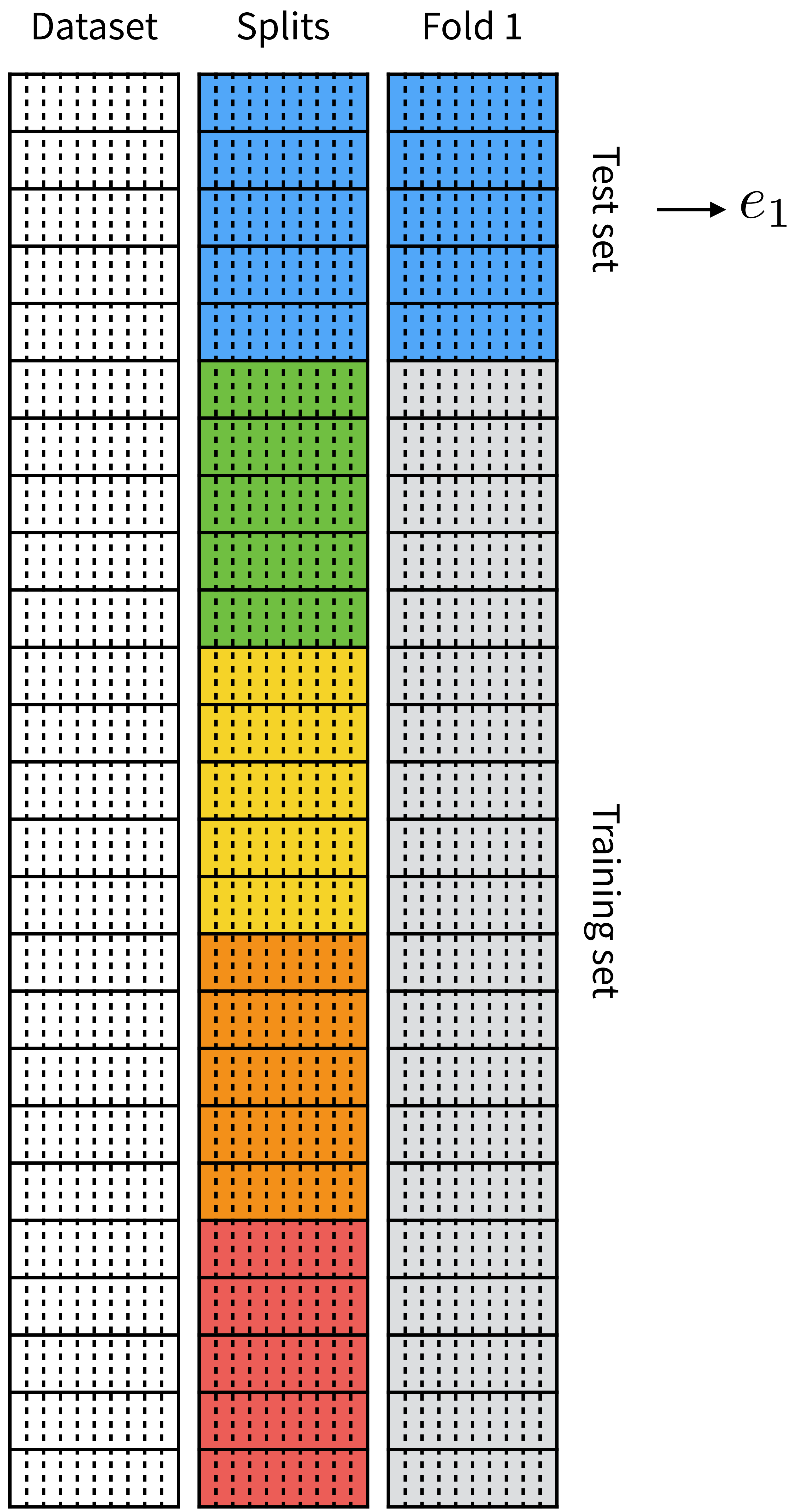
- Most machine learning algorithms have hyperparameters, settings that we use to control the algorithm's behaviour. These include:
  - Learning rate
  - Regularization (e.g. weight decay)
  - When to stop training (early stopping)
- Deep learning algorithms typically have more, associated with the model architecture, e.g.:
  - Number of layers
  - Number of hidden units in each layer
- How to set these?

Validation Sets

- A validation set is typically carved out of the training set for the purpose of choosing hyperparameters (e.g. ~20%)
- It is distinct from the test set
- It is important that the test set is not used in any way to make choices about the model, including the hyperparameters
- The validation error will typically underestimate the generalization error (because of its use to fit the model)

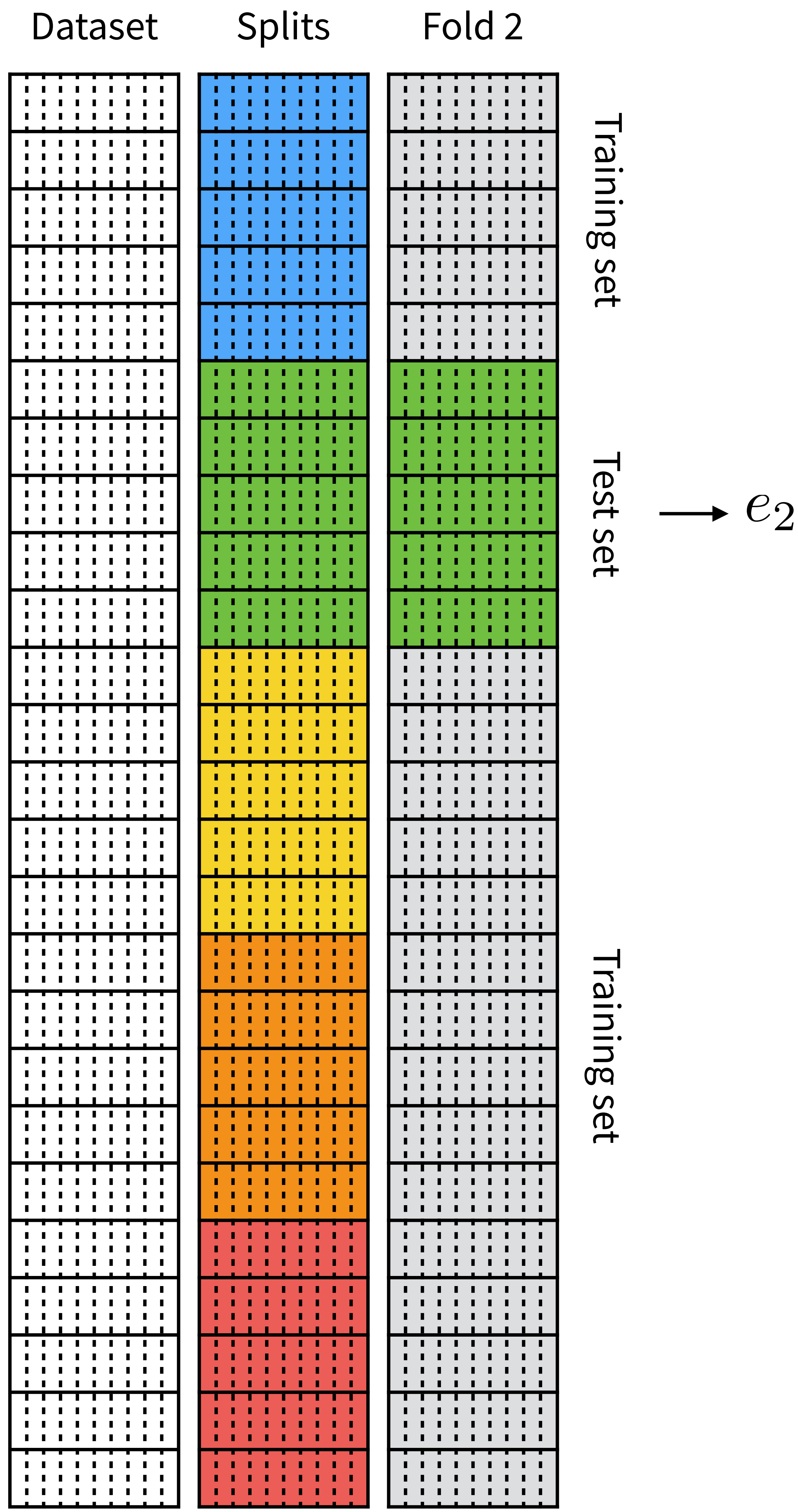
Cross-validation

- Dividing the dataset into a fixed training set and test set can be problematic if the dataset is very small
- Procedures based on repeating the training and testing computation on different randomly chosen subsets ("splits") allow one to use all the examples in the computation of mean test error
- The trade-off is computational complexity
- The most common such method is "k-fold" cross-validation (see 5-fold at right)



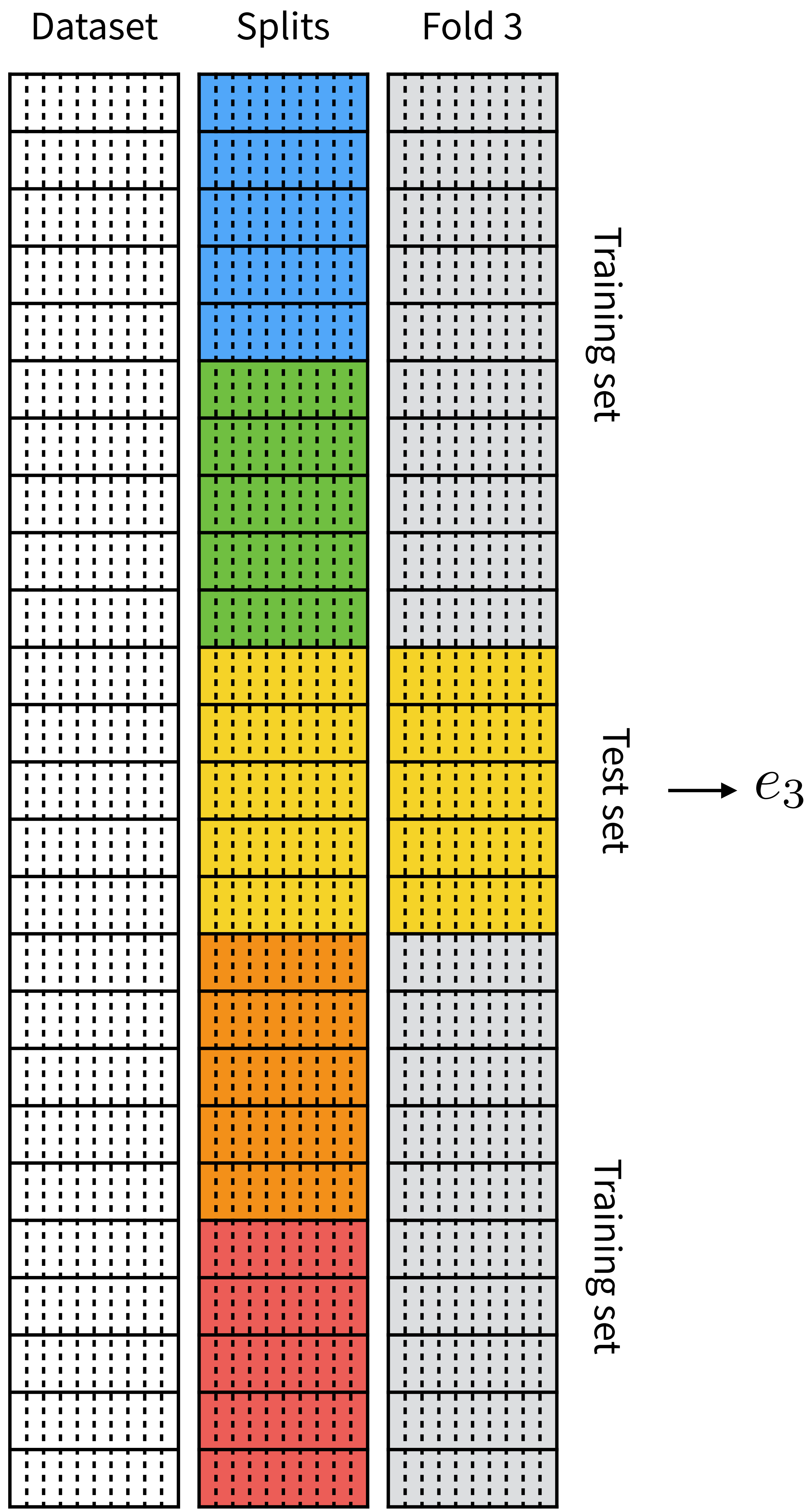
Cross-validation

- Dividing the dataset into a fixed training set and test set can be problematic if the dataset is very small
- Procedures based on repeating the training and testing computation on different randomly chosen subsets ("splits") allow one to use all the examples in the computation of mean test error
- The trade-off is computational complexity
- The most common such method is "k-fold" cross-validation (see 5-fold at right)



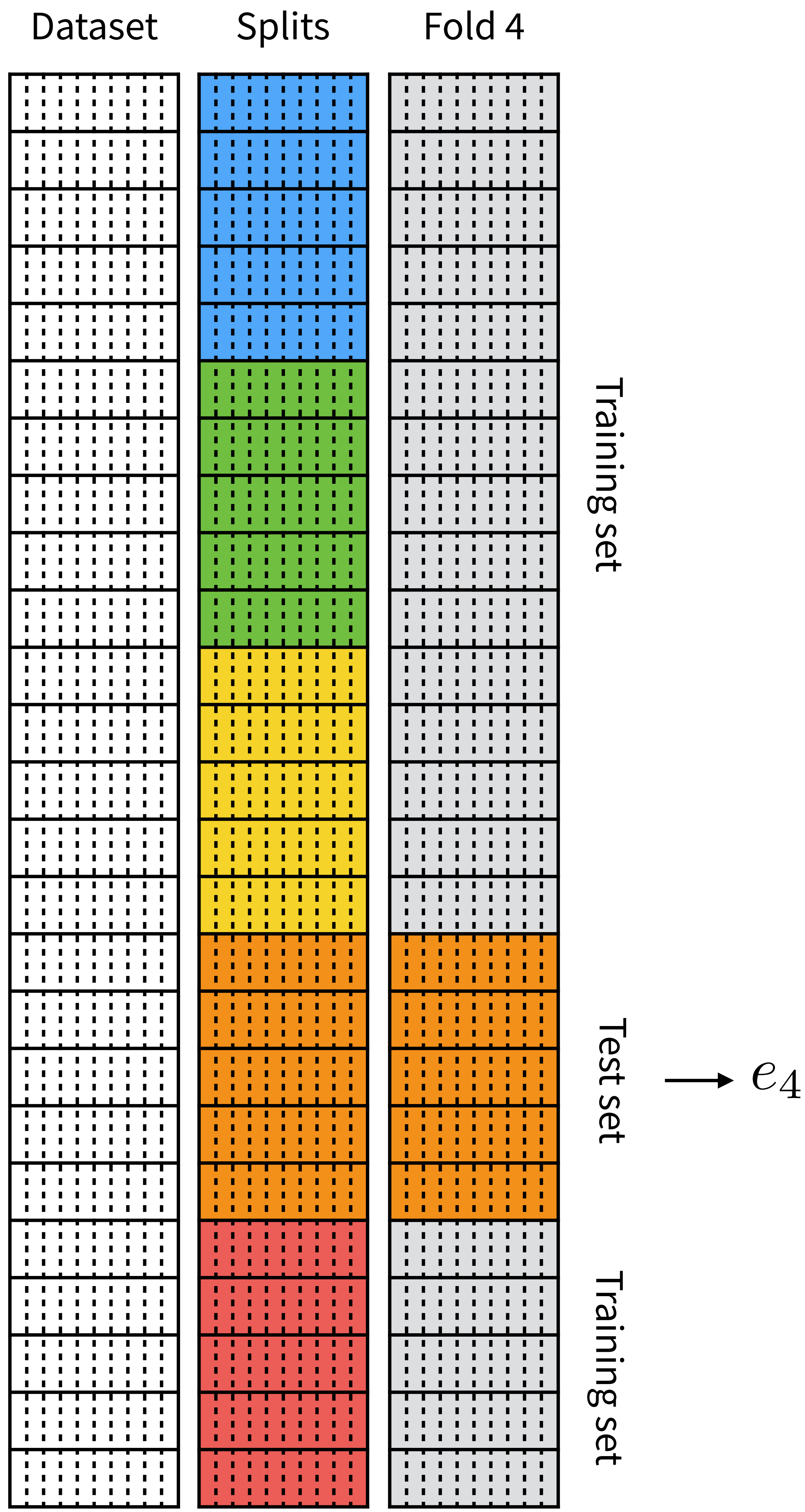
Cross-validation

- Dividing the dataset into a fixed training set and test set can be problematic if the dataset is very small
- Procedures based on repeating the training and testing computation on different randomly chosen subsets ("splits") allow one to use all the examples in the computation of mean test error
- The trade-off is computational complexity
- The most common such method is "k-fold" cross-validation (see 5-fold at right)



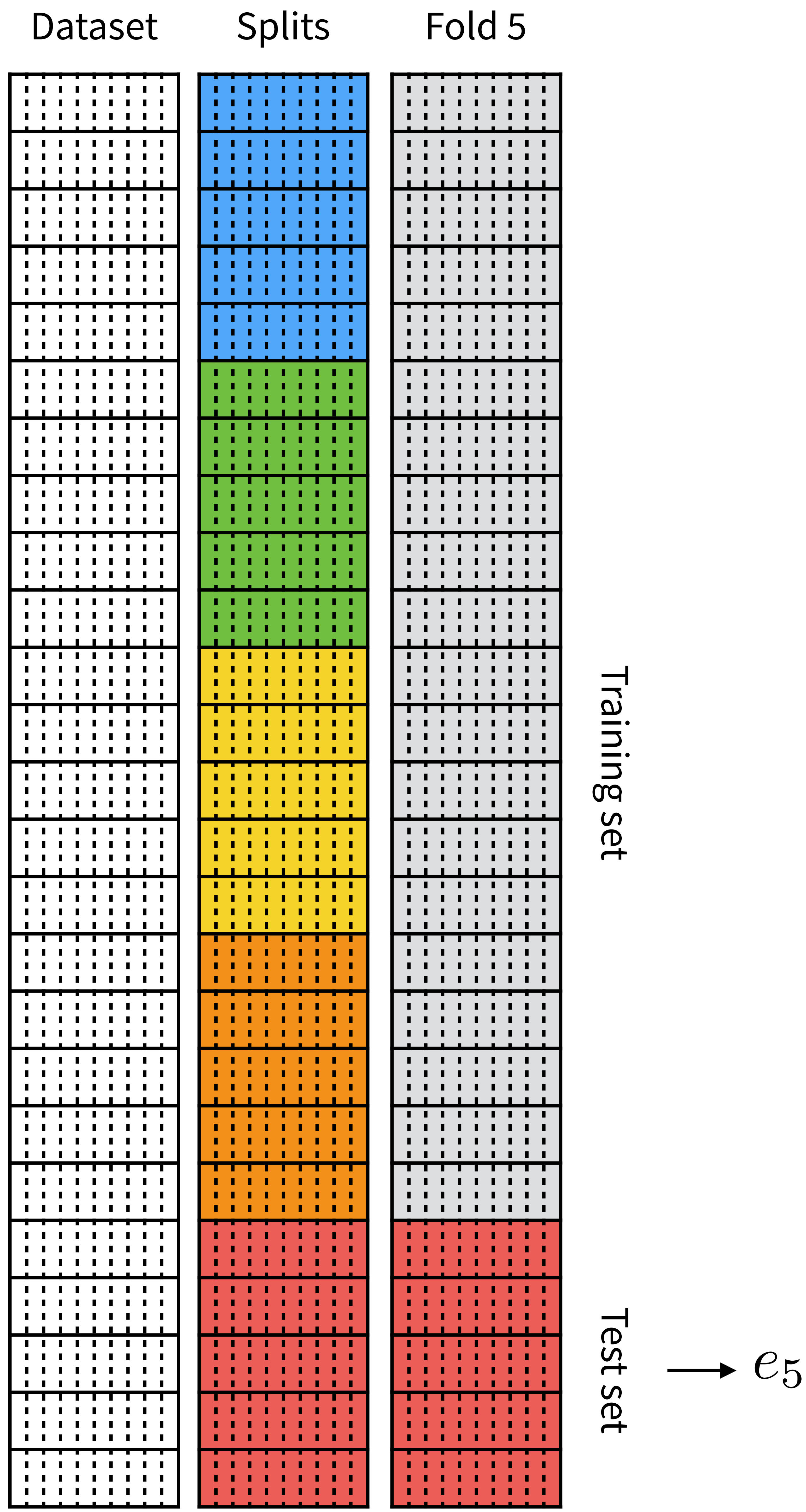
Cross-validation

- Dividing the dataset into a fixed training set and test set can be problematic if the dataset is very small
- Procedures based on repeating the training and testing computation on different randomly chosen subsets ("splits") allow one to use all the examples in the computation of mean test error
- The trade-off is computational complexity
- The most common such method is "k-fold" cross-validation (see 5-fold at right)



Cross-validation

- Dividing the dataset into a fixed training set and test set can be problematic if the dataset is very small
- Procedures based on repeating the training and testing computation on different randomly chosen subsets ("splits") allow one to use all the examples in the computation of mean test error
- The trade-off is computational complexity
- The most common such method is "k-fold" cross-validation (see 5-fold at right)



Finding Hyperparameters

Finding Hyperparameters

- Traditionally, there are two ways to set hyperparameters:

Finding Hyperparameters

- Traditionally, there are two ways to set hyperparameters:
  - Manually (e.g. by expert knowledge)

Finding Hyperparameters

- Traditionally, there are two ways to set hyperparameters:
  - Manually (e.g. by expert knowledge)
  - Automatically, by
    - systematic search, (e.g. grid, random)
    - Sequential model-based optimization

(nice entry point)

Shahriari et al. 2016. "Taking the Human Out of the Loop: A Review of Bayesian Optimization." Proceedings of the IEEE 104 (1): 148–75.

Automatic Hyperparameter Optimization Algorithms

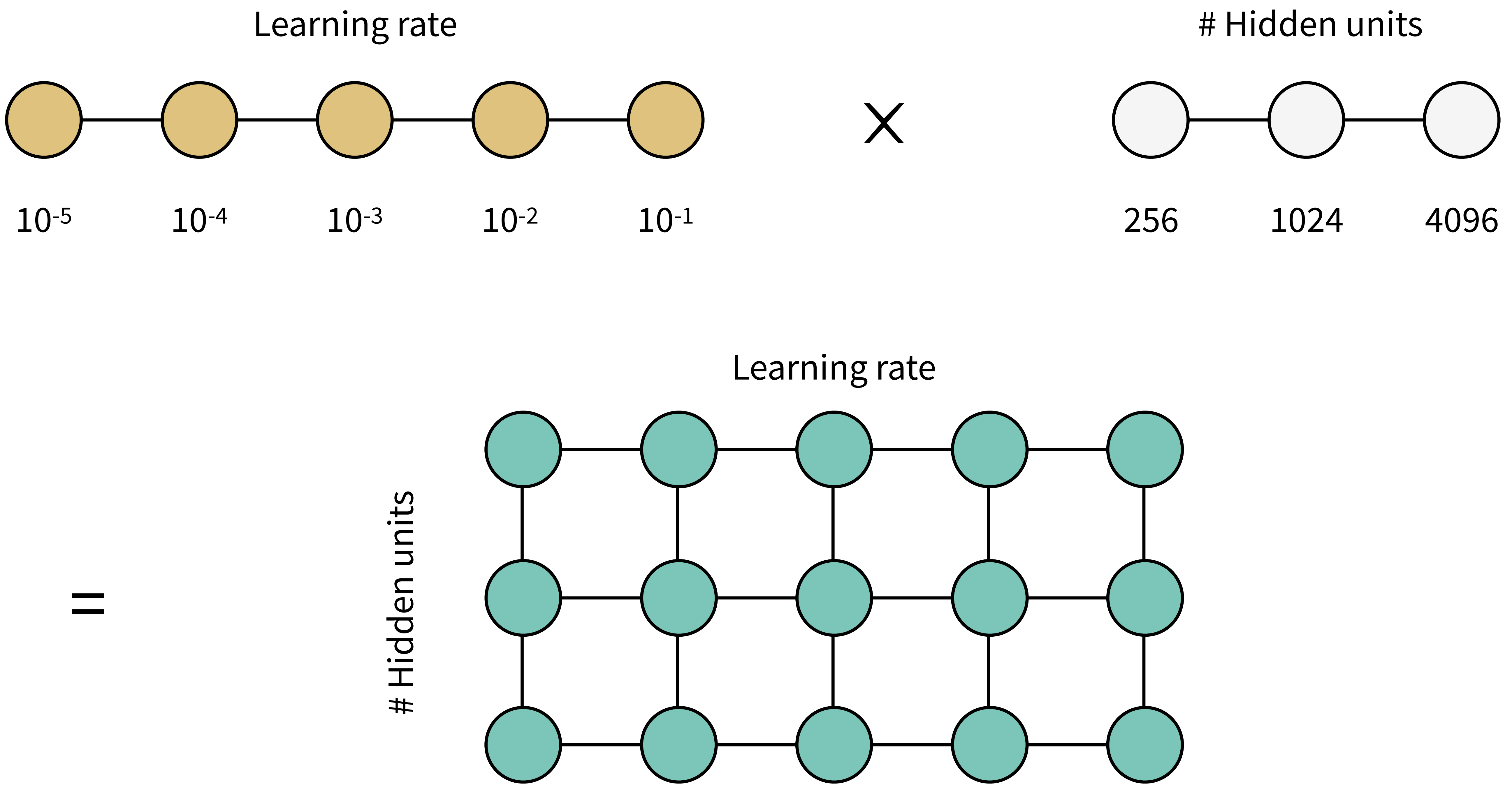
- The popularity of some learning algorithms (e.g. logistic regression, SVMs) stems in part from their ability to perform well with only 1-2 tuned hyperparameters

- Manual hyperparameter tuning can work very well when the user has a good starting point (e.g. baseline), or months or years of experience

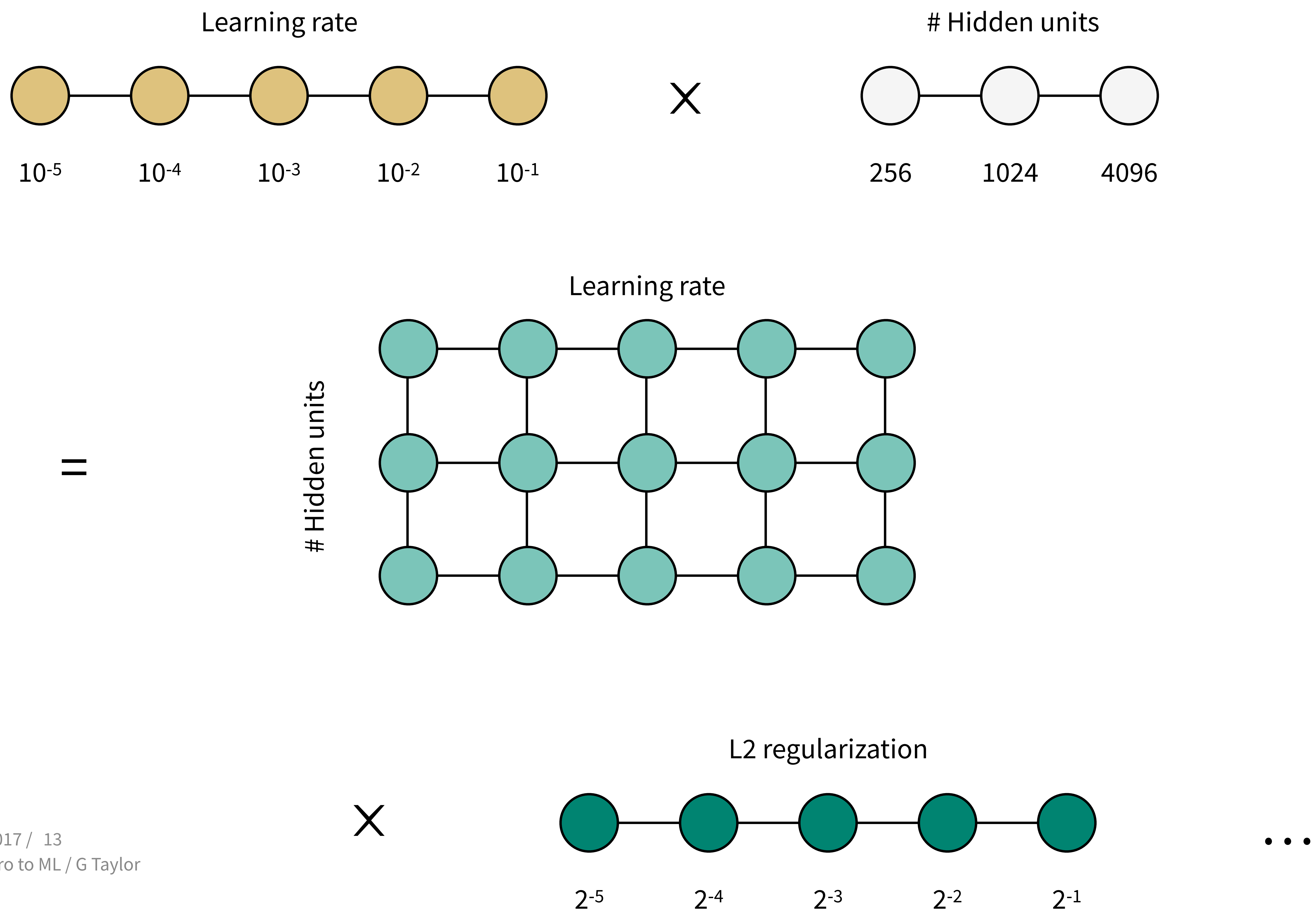
- For many applications, these starting points are not available, so we turn to automated methods

- Caution: these algorithms have their own hyperparameters

Grid search


Learning rate
10-5
10-4
10-3
10-2
10-1
×
Hidden units
256
1024
4096
=
Learning rate
Hidden units

Grid search



Learning rate
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
×
Hidden units
256
1024
4096
=
Learning rate
Hidden units
×
L2 regularization
2^{-5}
2^{-4}
2^{-3}
2^{-2}
2^{-1}
…

Random search

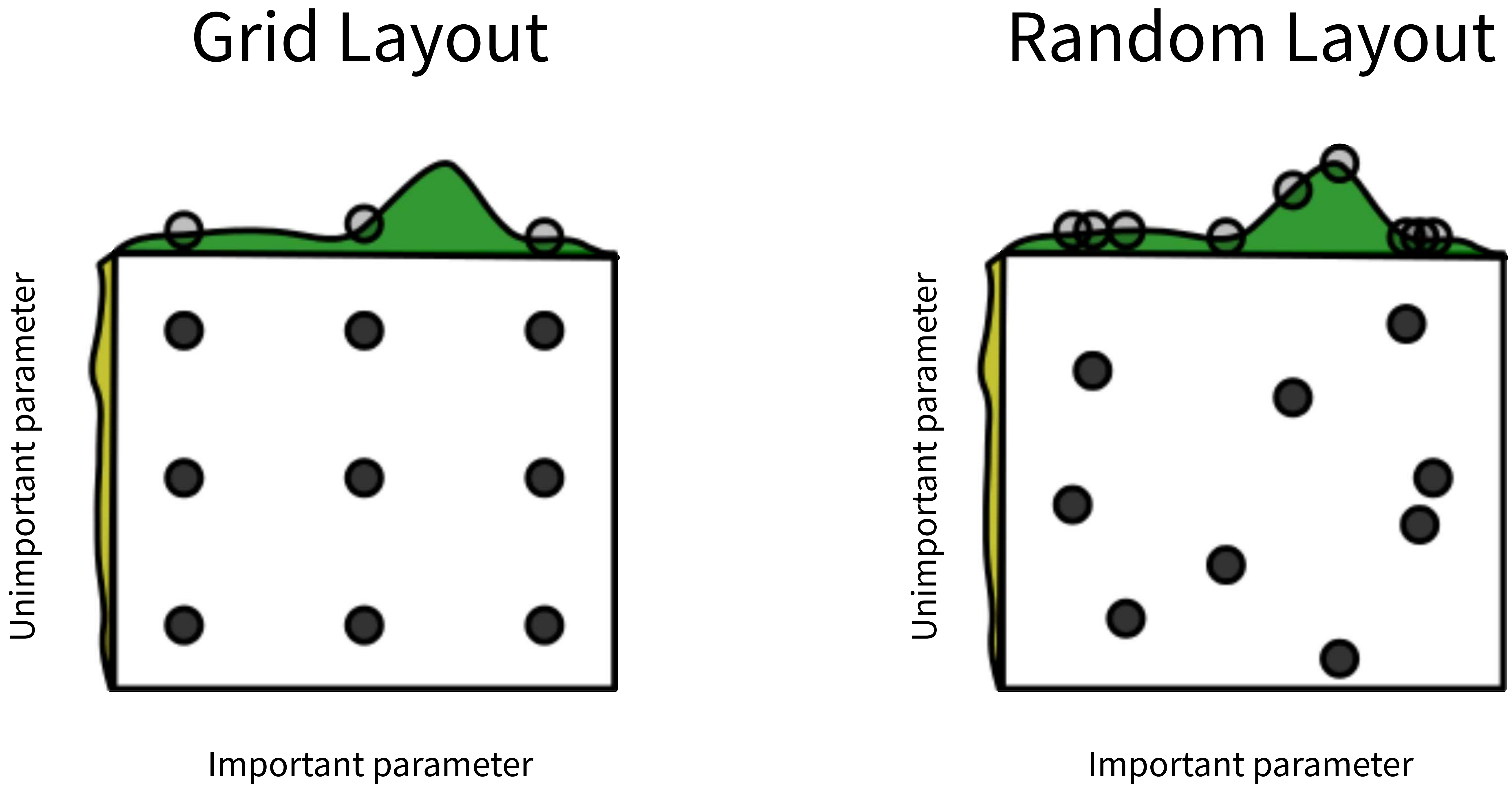


Image Credit: Bergstra and Bengio (2012)

Bayesian optimization

t=2

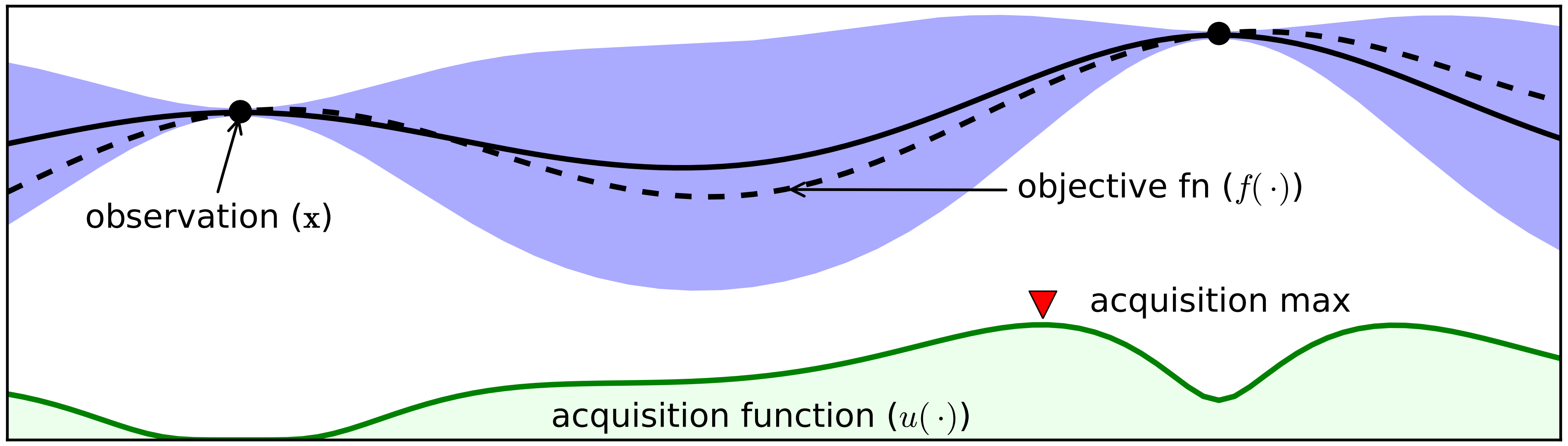


Image Credit: Brochu et al. (2010)

Bayesian optimization

t=2

t=3

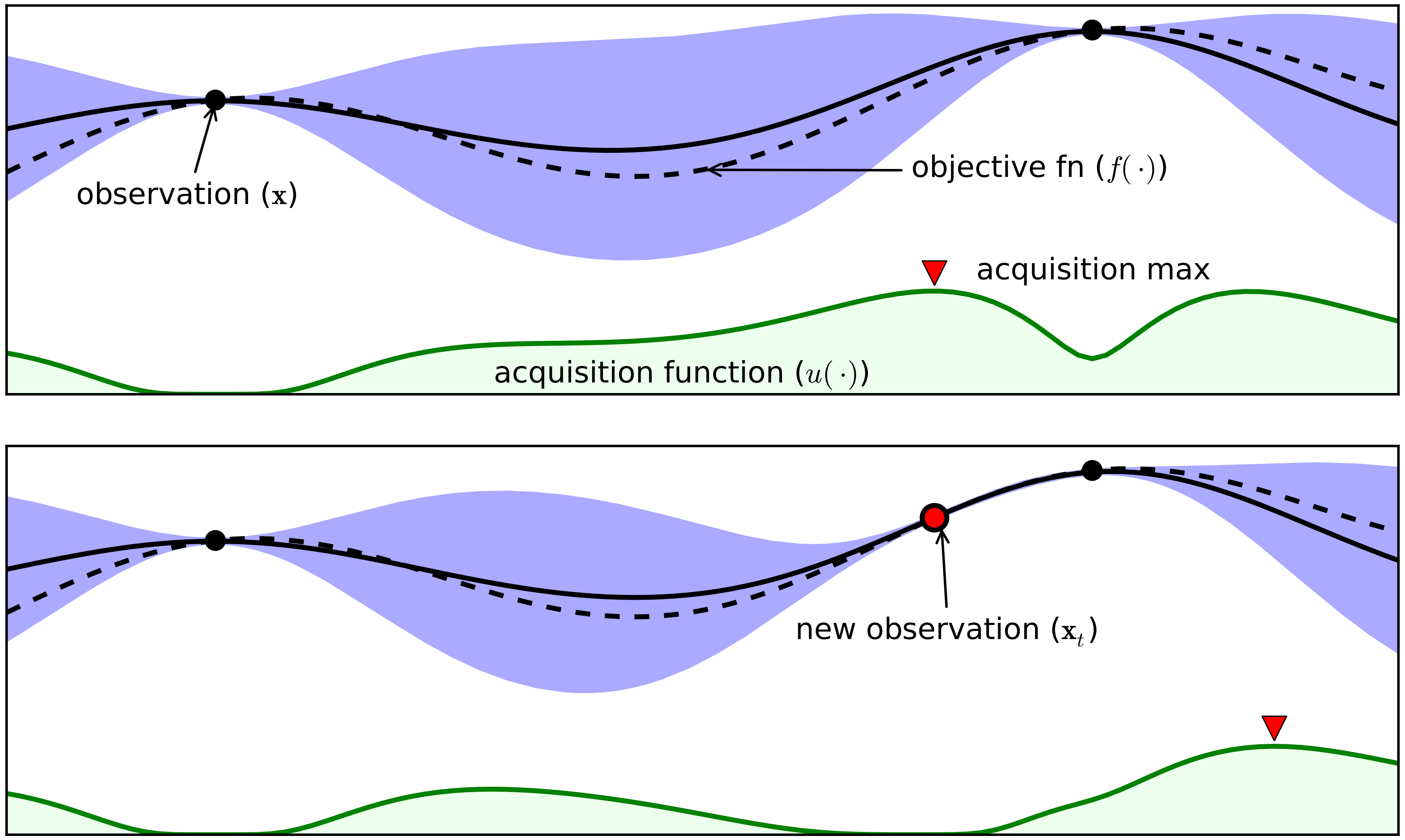


Image Credit: Brochu et al. (2010)

Bayesian optimization

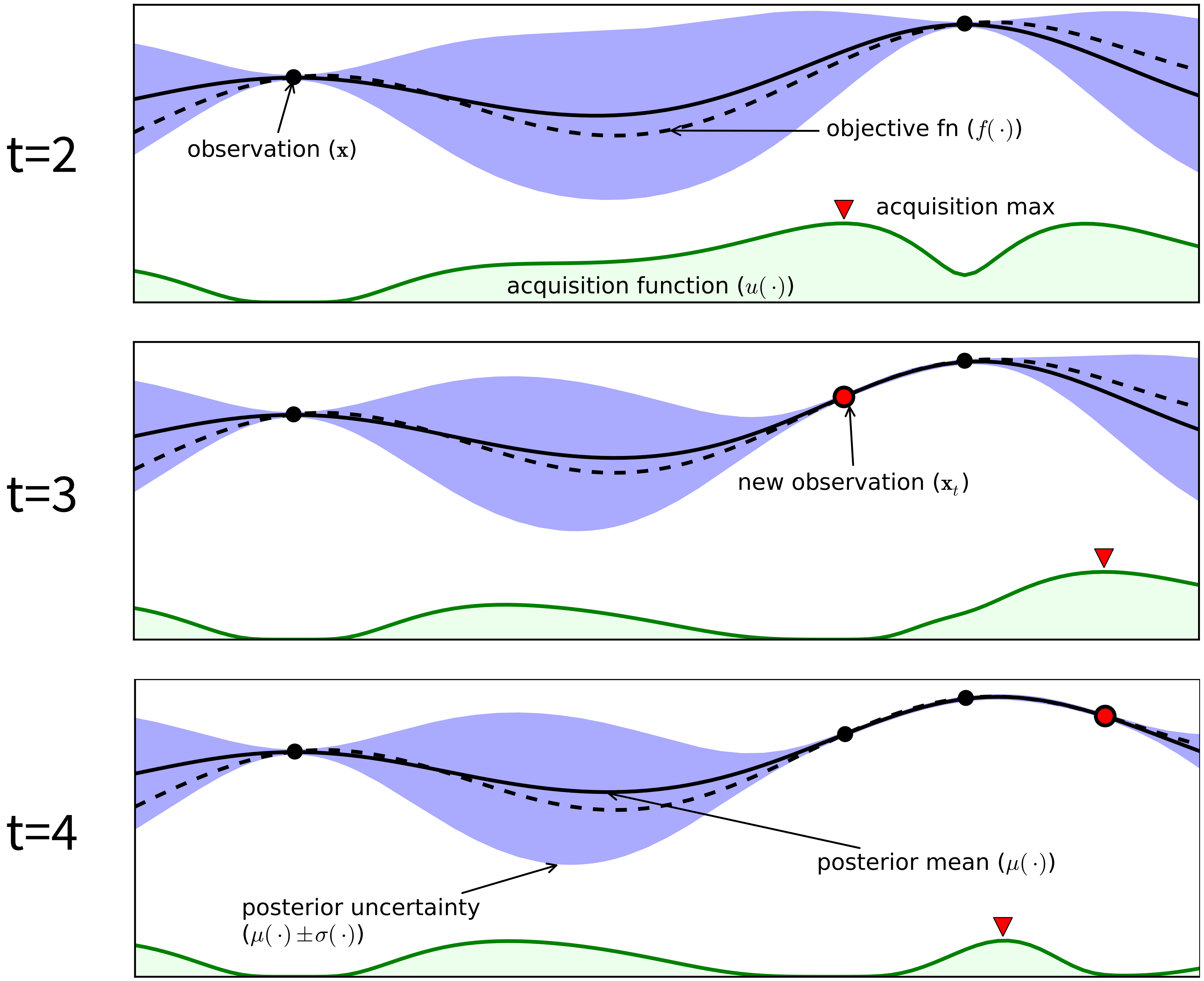


Image Credit: Brochu et al. (2010)

Advice

Advice

- For simple models with < 4 hyperparameters, use grid search

Advice

- For simple models with < 4 hyperparameters, use grid search

- For deep learning (typically many more hyperparameters) try random search first

    - If you're experienced and feeling adventurous, try Bayesian optimization

    - Monitor architecture and optimizer learning research (e.g. Zoph and Le 2017)