Robust Evaluation of Machine Learned Models

Andrew H. Fagg
Symbiotic Computing Laboratory

Goals for Building Reliable Models

We want:

- Models that will work well with future data
- A sense of how sensitive our model performance is to:
  - The specific training data that we are using
  - The amount of training data that we have
- Formal ways of selecting model hyper-parameters
- Formal ways of comparing two (or more) different model types (the bake off!)

Definitions

- **Parameters**: parameters that are selected by a learning algorithms
- **Hyper-parameters**: parameters that are selected outside the learning algorithm, but affect how it behaves
  - Regularization
  - Structure: number of layers, number of computing elements within a layer, …
- **Model type**: broad category of models (e.g., deep network vs a support vector machine)

Data Universe

Contains all possible data samples, including all time

A First Approach

Ideal process:

- We can observe the entire data universe
- Construct a model that explains all of these samples
- Done

A First Approach

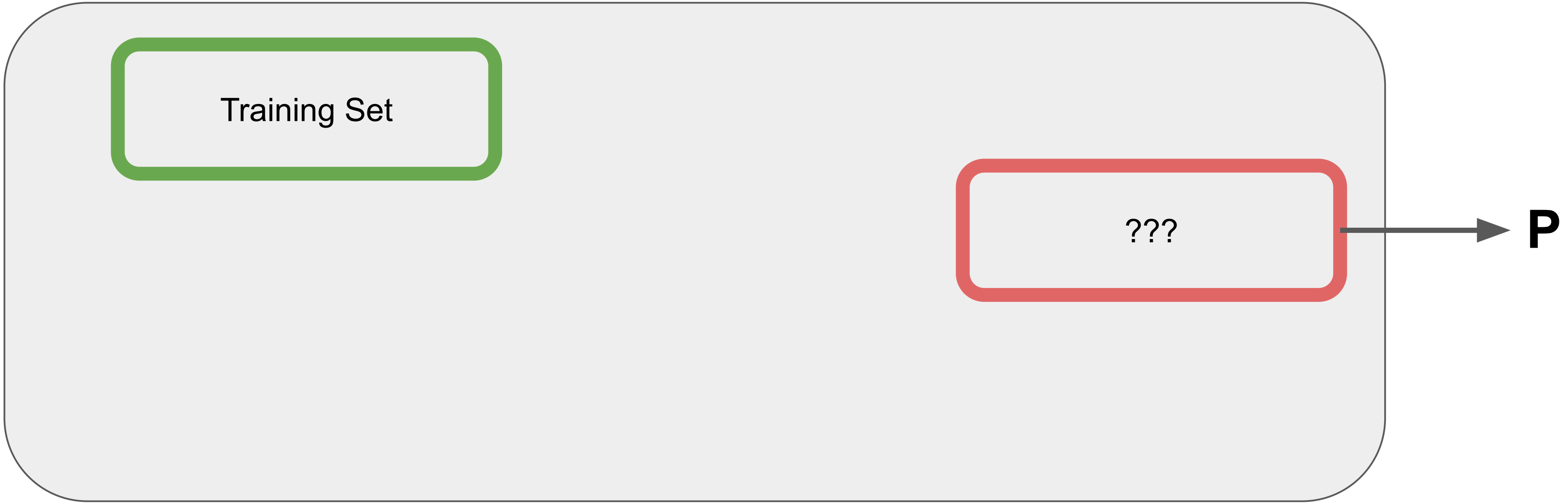
Challenges:

- Sampling the universe is typically not feasible
- Even when we can sample the universe, it may not be feasible to use with our learning algorithm

A Second Approach

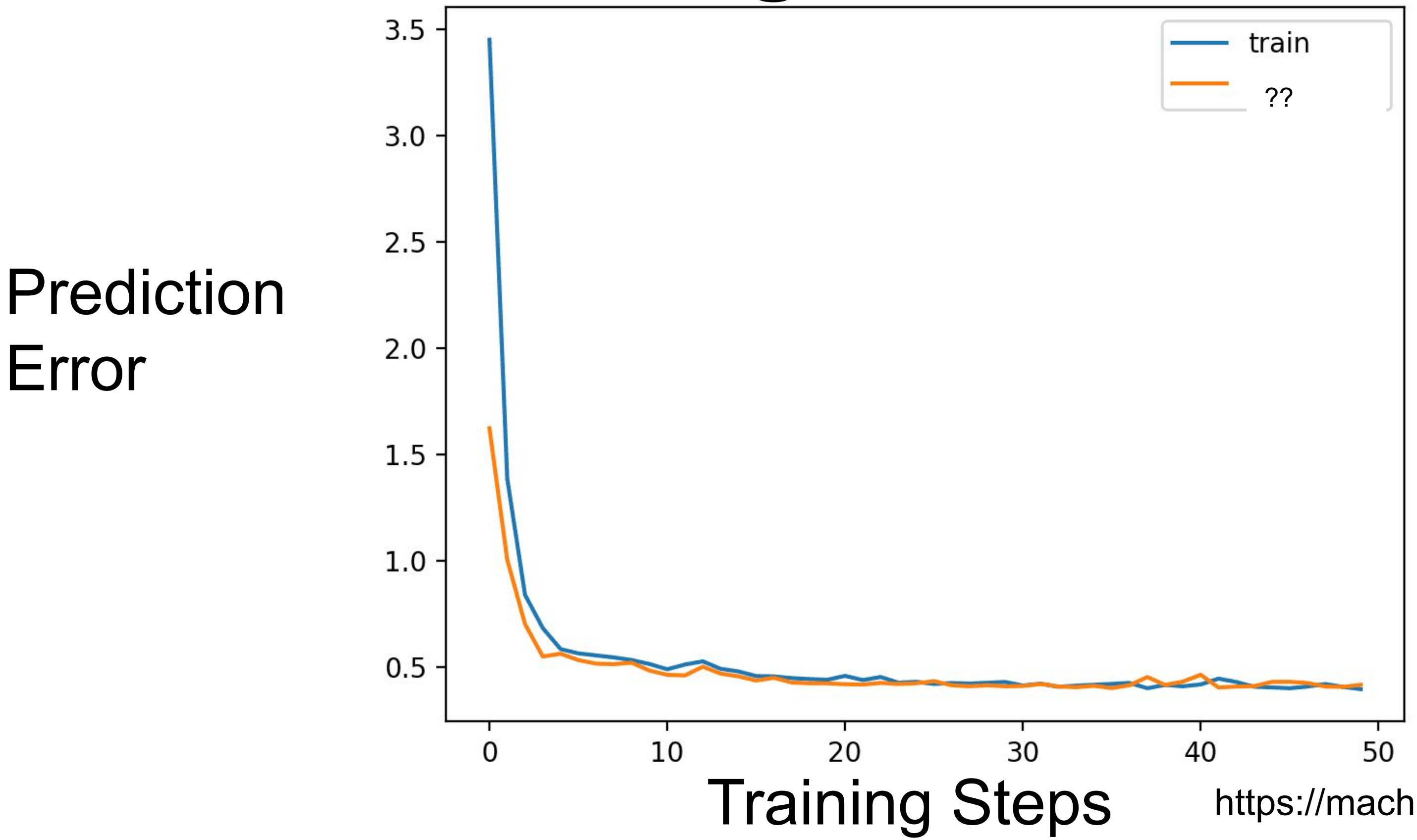
Take a sample from the universe for the purpose of training the model

A Second Approach



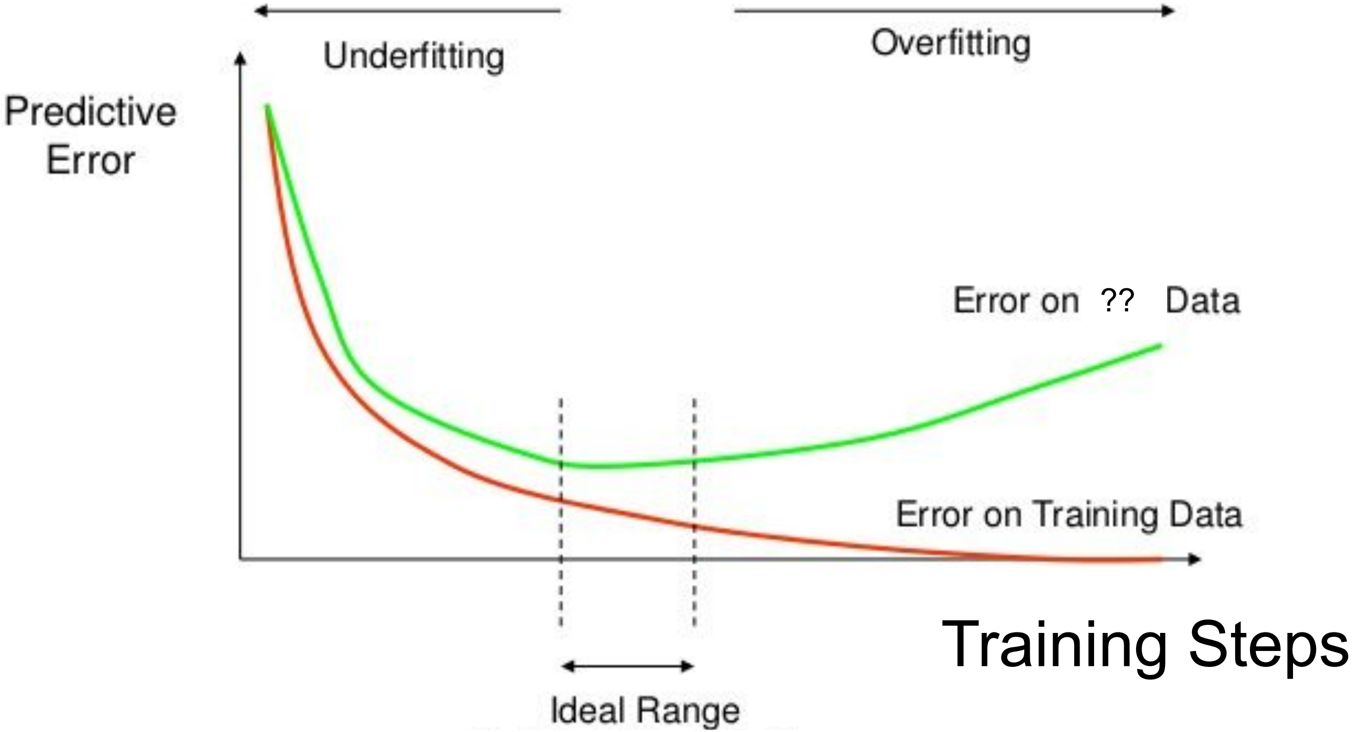
P is the performance of the learned model on independent data set

Learning Curves: Ideal



https://machinelearningmastery.com

Learning Curves: Overfitting



https://i.stack.imgur.com/rpqa6.jpg

Combatting Overfitting

- Increase training set size
- Reduce number of parameters
- Regularization techniques: force simpler models
  - Explicit: add model complexity to cost function
  - Implicit: random reduction in model complexity (e.g., dropout)
- Early stopping: use independent data set performance to halt the gradient descent process

Building a Model is a Stochastic Process

- Sampling a data set from the universe
- Learning algorithms often have stochastic elements
  - Initial parameter choices are often selected from a distribution
  - Approximate gradient descent using a subset of training set examples
  - Sampling question types in a decision tree

Building a Model is a Stochastic Process

- We need to treat all performance measures as random variables
- So, a single observation is not sufficient to conclude anything, especially if we want to formally compare model types
- And - we need a sufficient number of observations to apply our hypothesis testing tools

A Third Approach

Statistically independent training and evaluation data sets

A Third Approach

- N performance measures are also statistically independent
- Can treat as a set of IID samples from a distribution. Can then answer:
  - Did we learn anything?
  - How does this model type compare with another model type?
    - If we use the same training / evaluation data set pairs, then can use a paired statistical test

Challenges of the Third Approach

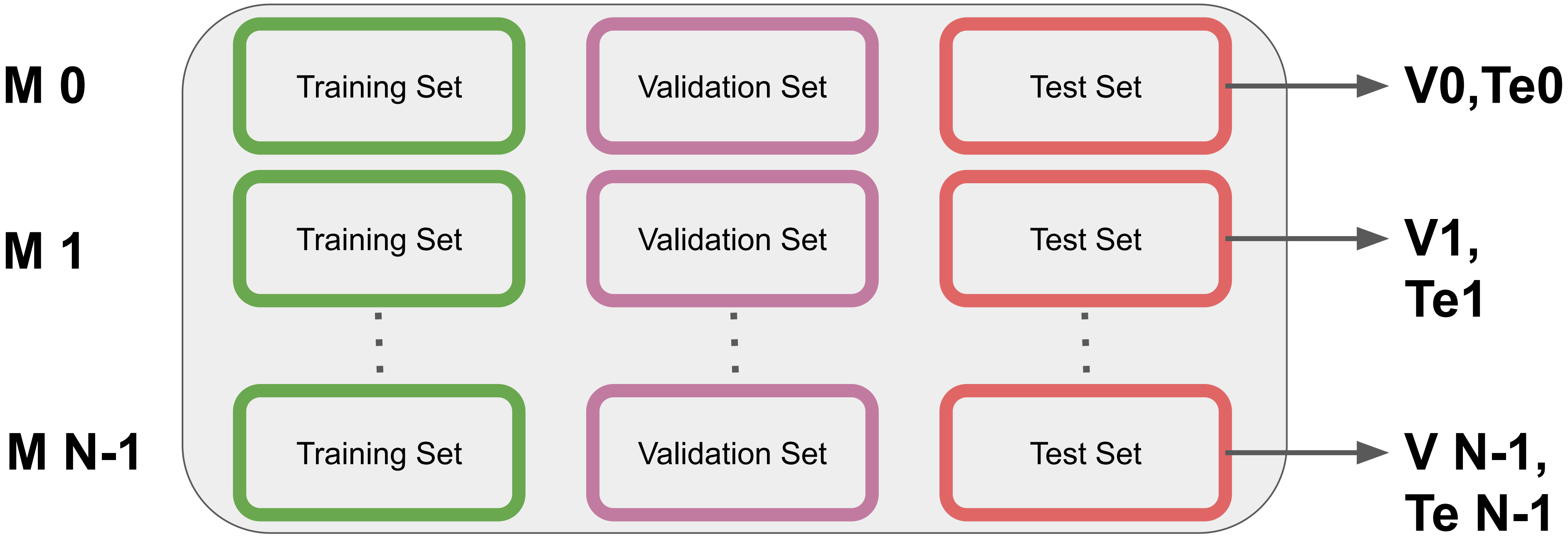
- Each model type has many possible hyper-parameters
- Before we compare model types, we need to choose the appropriate hyper-parameter set for each
- But, how to make this choice?
  - Hyper-parameters often affect the degree of overfitting, so training set data cannot be used to make this choice
  - But, using the same Test Data Set, we run the risk of overfitting the hyper-parameters to this data set

Three Different Data Set Types

Data sets that are IID:

- **Training set**: used by the learning algorithm to select parameters
- **Validation set**:
  - Select a stopping point for training
  - Select hyper-parameter choices
- **Test set**:
  - Reporting results
  - Formal comparison between model types

A Fourth Approach



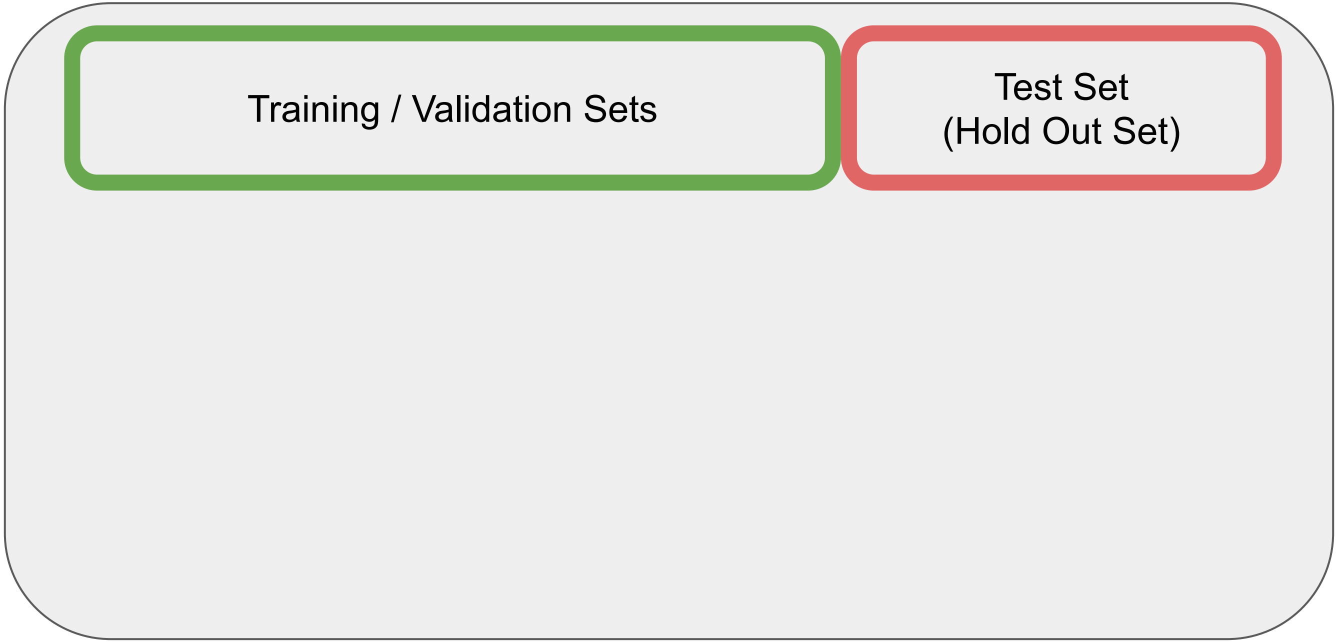
Specifics

- Use Vi to determine the end of the training process for model i
- Use average (V) to compare different hyper-parameter choices
- Use Te i’s for final evaluation and comparison between types of models (the bake-off)

Challenges

- So far, we have assumed that sampling from the universe is easy / inexpensive
- In practice, this is not the case:
  - Real limitations in our ability to collect / label data
- But still want sound approaches to:
  - Selecting hyper-parameters
  - Comparing model types
- And: want some way to understand sensitivity of model performance with respect to training set size

A Fifth Approach:
N-Fold Cross-Validation



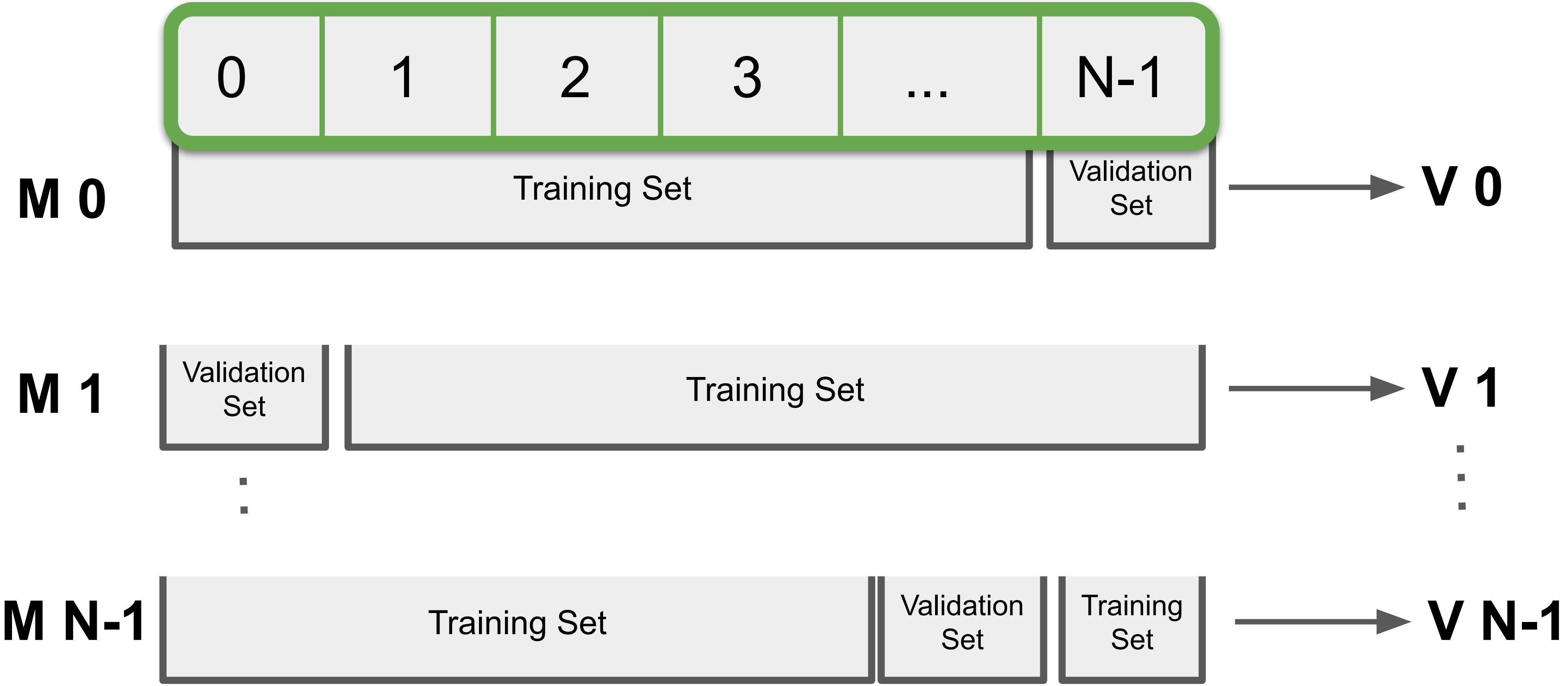
Sample what we can from the universe & split into two pieces

N-Fold Cross-Validation

Cut training/validation set into N independent folds

Construct N different models with different subsets of the folds

N-Fold Cross-Validation



Specifics

- A single sample occurs in exactly one fold
- Use Vi to determine the end of the training process for model i
- Use average(V) to compare different hyper-parameter choices
  - Choose the hyper-parameter set with the best average(V)
  - Call this H*

Evaluating

Comparing model types:

- Evaluate each of the N models with the same Test Data Set
- This gives us N metrics: Te 0, Te 1, … Te N-1
- Do the same for another model type:
  - Te 0’, Te 1’, … Te N-1’
- Use hypothesis testing to compare these two distributions

Reporting Performance / Future Use

- Use all N folds to train a new model using hyper-parameters H*
- Evaluate this new model using the Test Data Set. Report this performance
- Use this model with future data

N-Fold Cross-Validation

Dominant approach

- Many papers, blog posts, books
- Built into standard toolkits, including SciKit Learn (e.g., cross_val_predict())

But there is a problem...

N-Fold Cross-Validation

But there is a problem…

- Because the same Test Data Set is used to compute all N performance measures (Te 0, Te 1, … Te N-1), they are not independent from one-another
- This precludes our use of many standard hypothesis testing tools

N-Fold Cross-Validation

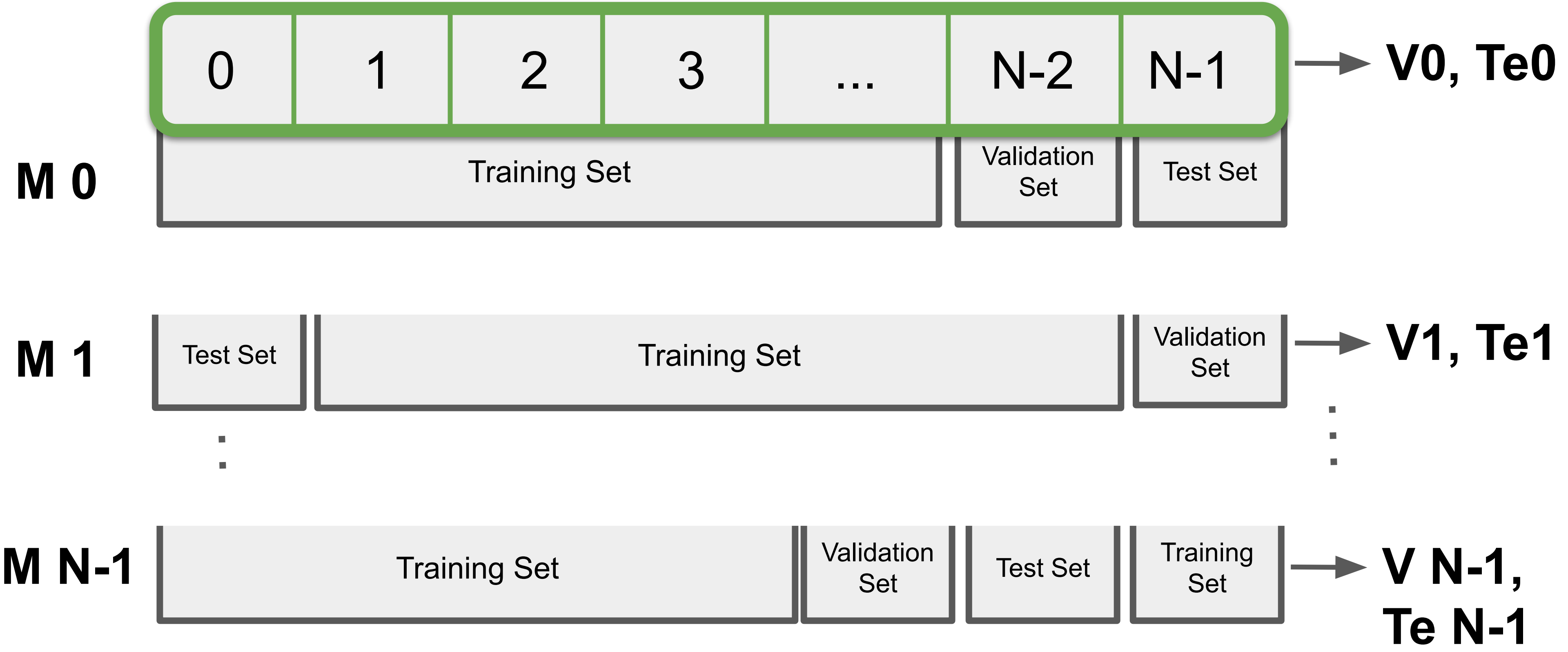
How do we repair this?

- Could cut the Test Data Set into N independent folds and use a different one to evaluate each of the N models
  - Potentially increase the variance of the performance metrics (especially a problem if the Test Data are already small or sparse)
- Draw the Test Data Set from the original set of folds

“Holistic” N-Fold Cross-Validation

- Cut available data into N independent folds
- For each model, use
  - N-2 folds for training
  - 1 fold for validation
  - 1 fold for testing

Holistic N-Fold Cross-Validation



Specifics

- Use Vi to determine the end of the training process for model i
- Use average(V) to compare different hyper-parameter choices
    - Choose the hyper-parameter set with the best average(V)
    - Call this H*
    - Note: we are not allowed to look at Te 0 … Te N-1
        - Right now, we just cache these performance metrics

Evaluating

Comparing model types:

- For model type 1:
  - We have identified H*
  - Extract from the cached Te 0 .. Te N-1 for H*
- For model type 2:
  - H*’ gives us Te 0’ … Te N-1’
- Use hypothesis testing to compare these two distributions

Notes

- A single data set example is used exactly once for validation and once for testing
- If the samples are independent, this means that our test folds are independent from one-another
- This means Te 0 … Te N-1 are independent (maybe)
- So, can use our standard hypothesis testing tools

Caveats

- Holistic cross-validation uses one less fold for training than cross-validation
  - But does not require a hold-out set
- In either case, the training sets are **NOT** independent
  - Means that the models themselves are not independent
  - So… Te 0 .. Te N-1 may not be truly independent
  - In practice, if the folds individually reflect the distribution of the universe, then this is probably not a problem (will return to this)

Sensitivity to Training Set Size

https://www.practicalai.io/how-to-debug-and-diagnose-machine-learning-problems/

Sensitivity to Training Set Size

https://www.practicalai.io/how-to-debug-and-diagnose-machine-learning-problems/

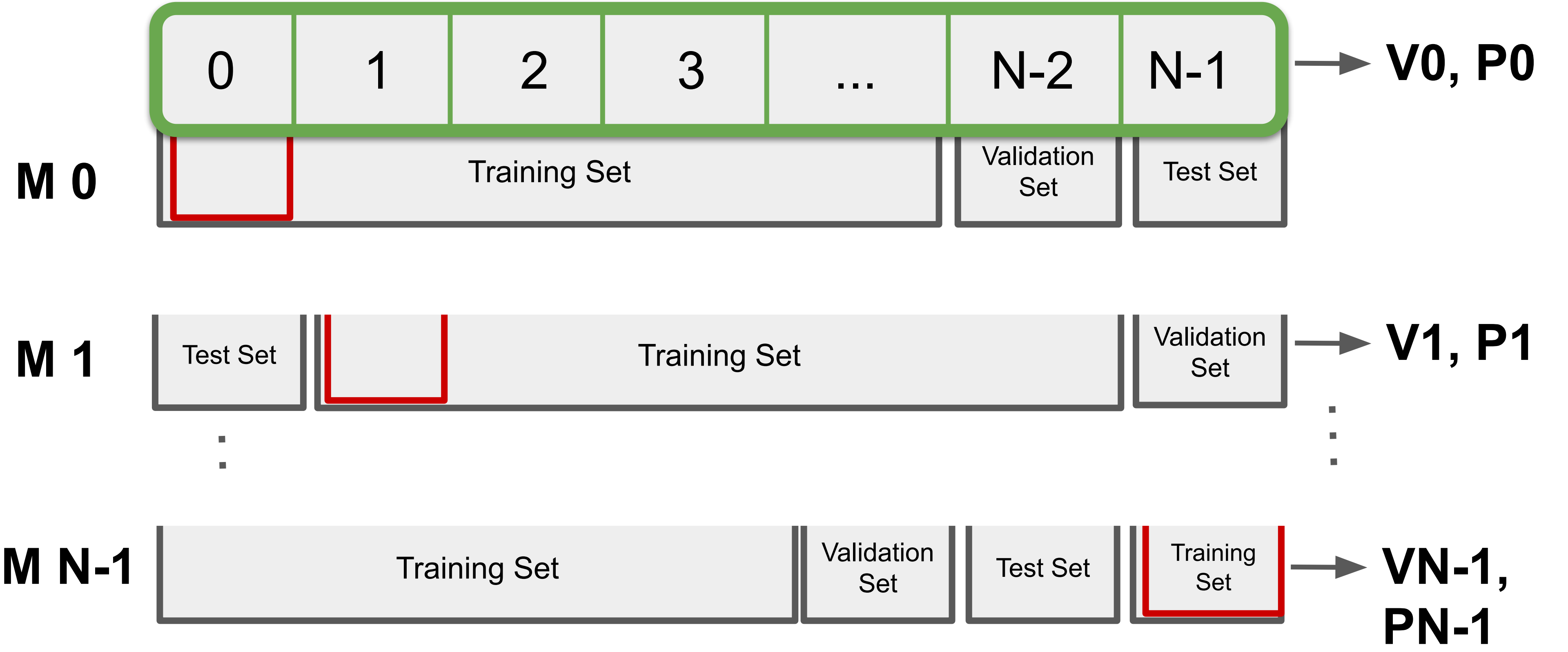
Sensitivity to Training Set Size

- Want to understand how sensitive a model type is to training size
- We might choose different hyper-parameters for different sizes

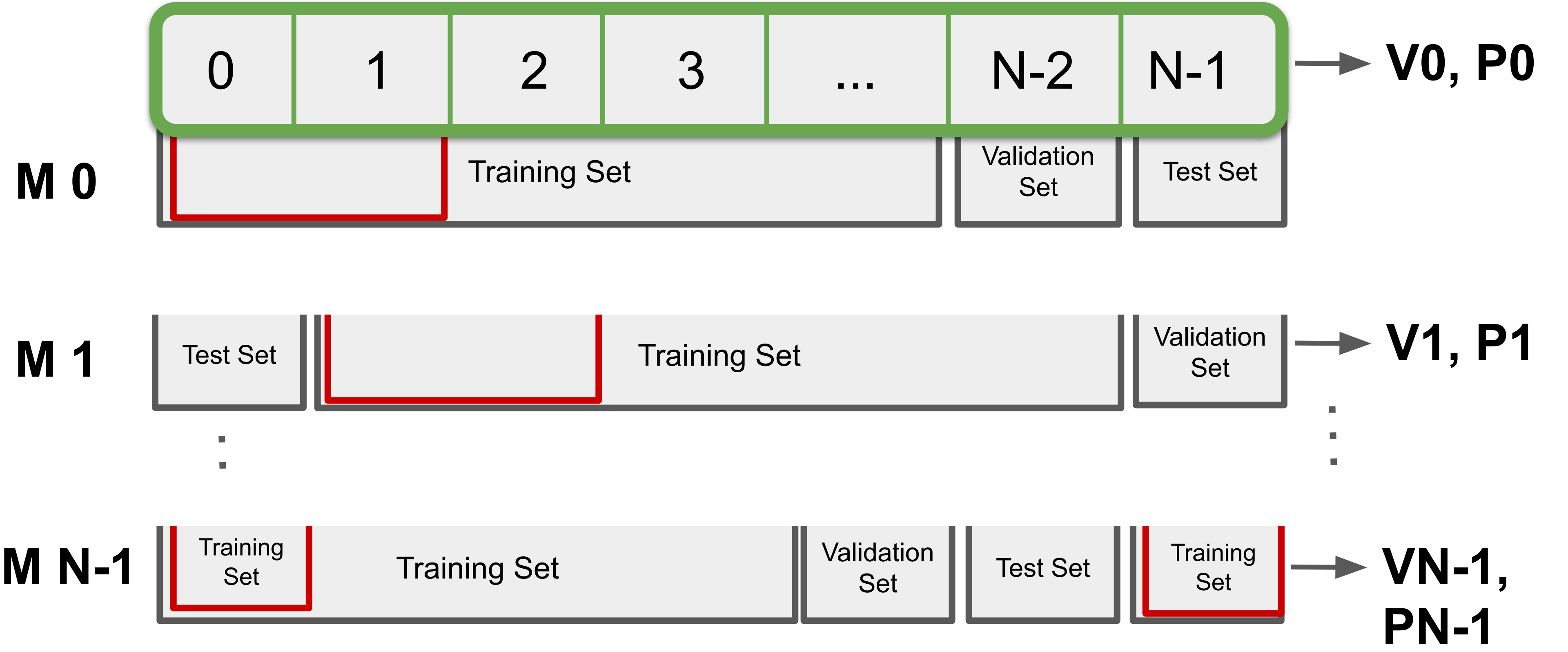
Implementation with Holistic N-Fold Cross-Validation

- Use only k of the available N-2 training folds
- These k rotate with the validation and test data sets

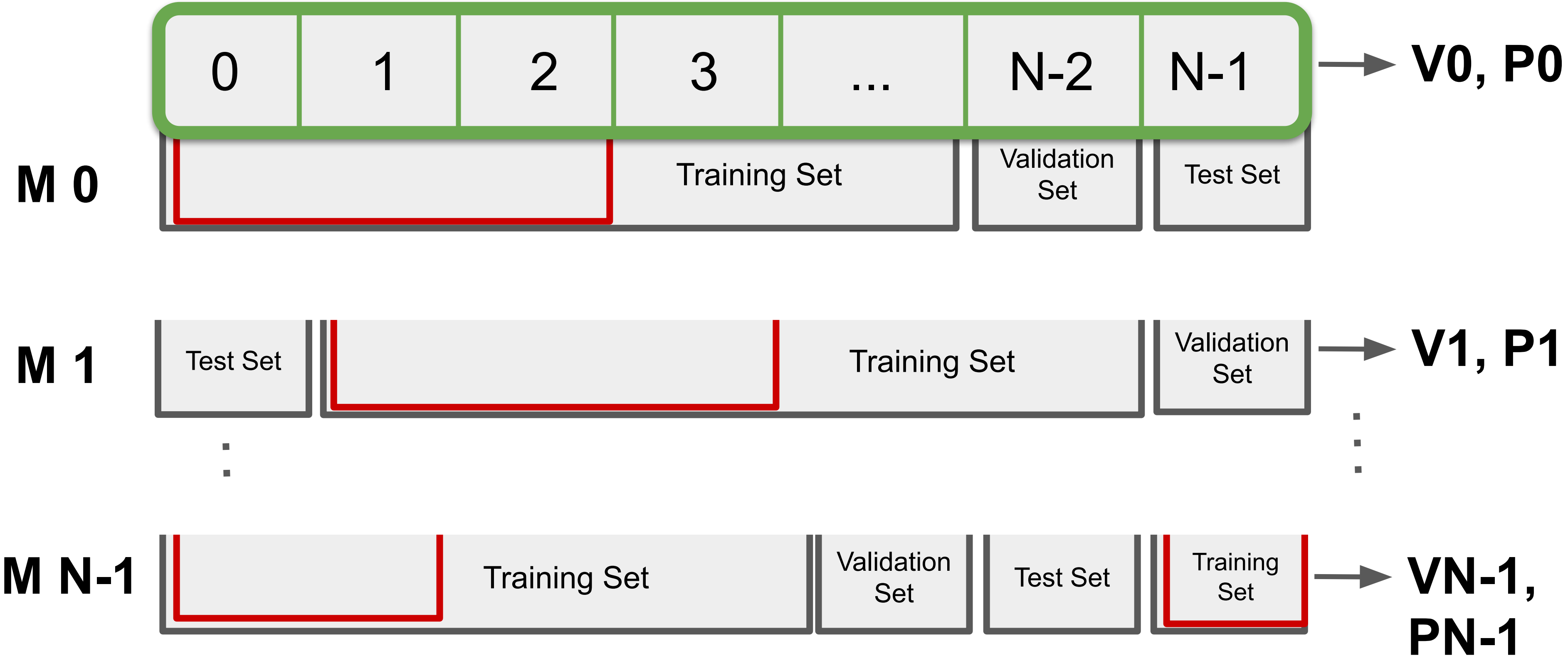
Training Size: 1 Fold



Training Size: 2 Folds



Training Size: 3 Folds



A Little Code ...

- nfolds = Total number of folds
- trainsize = Number of folds used for the training set:
  1, 2, … nfolds-2
- rotation = one of: 0, 1, … nfolds-1

```
trainfolds =    (range(trainsize) + rotation) % nfolds
valfold =       (nfolds - 2 + rotation) % nfolds
testfold =      (nfolds - 1 + rotation) % nfolds
```

Details

- For a single model type, a total of N x M x L models are learned & evaluated
  - N folds (so, N rotations) **20**
  - M hyper-parameter sets **???**
  - L choices for training set size **factors of 2**
- We typically reserve this process for formal evaluation
- And: do a lot of informal work ahead of time to explore hyper-parameter possibilities and training set sizes

Practicalities

- How small can we make N?
  - 20-30 is nice; 10 is not uncommon; 5?
- Training data set size sensitivity analysis is often done informally
  - Interacts with hyper-parameter selection

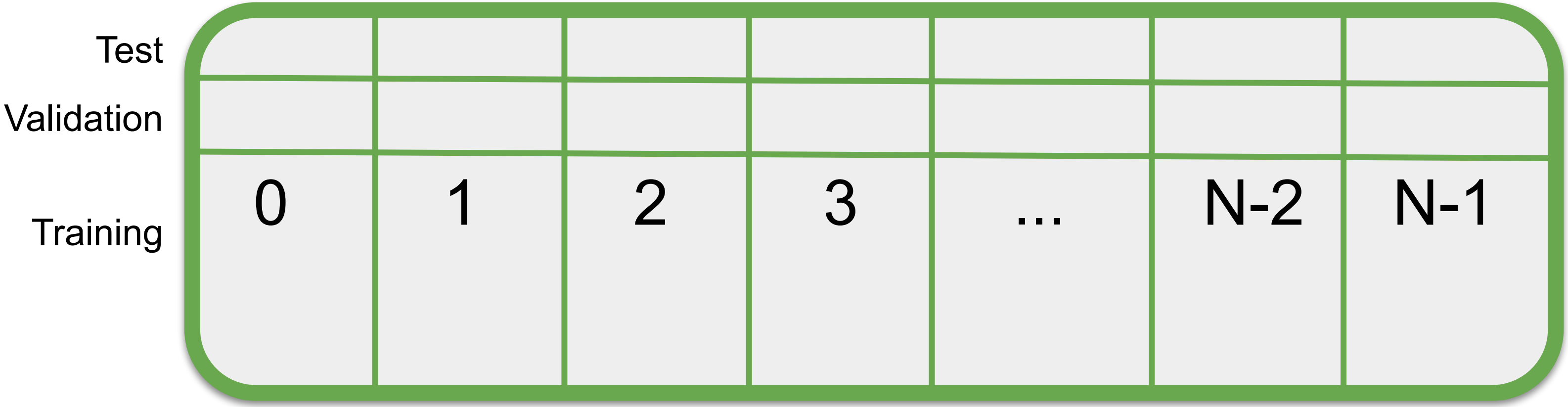
Challenges with Holistic Cross-Validation

- For a given rotation, the testing fold is independent of the training and validation folds
- However, to make decisions about hyperparameters, we don’t look at validation performance for a single rotation, but the mean across all validation folds
- One can argue that because validation fold for rotation k+1 (mod N) is the same as test fold for rotation k, that the performance measures are not truly independent

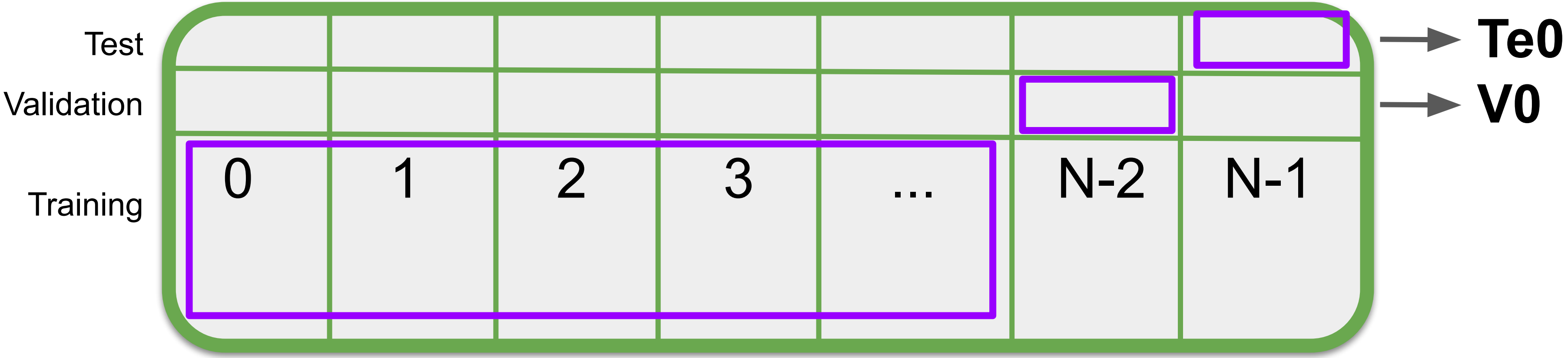
Orthogonal Cross-Validation

- Goal: fully independent validation and testing measures while using as much of the data for any rotation as possible
- Variety of solutions
- One approach:
  - Hold-out sets for both validation and testing
  - Cut each hold-out set into N folds

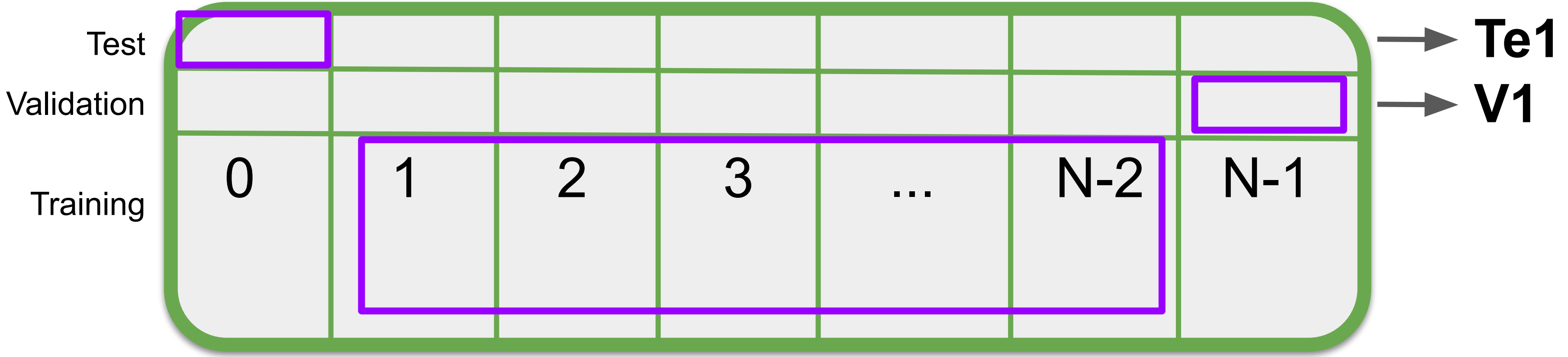
Orthogonal N-Fold Cross-Validation



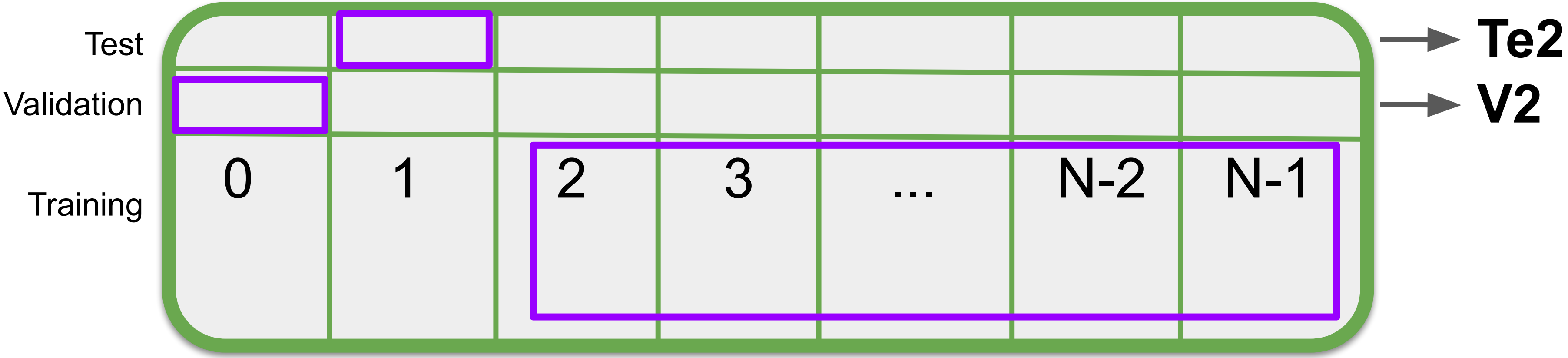
Orthogonal N-Fold Cross-Validation Rotation 0



Orthogonal N-Fold Cross-Validation Rotation 1



Orthogonal N-Fold Cross-Validation Rotation 2



Orthogonal N-Fold Cross-Validation

- For each rotation
  - Leaving some of the available data untouched
  - Using less data for training, validation and/or testing
- But, we feel more confident in the independence of the testing measures

Take-Aways

- Statistical evaluation matters
- There isn’t one solution to this
- Don’t confuse validation and test data sets
    - Can’t look at test data performance until the very end (though it is often convenient to compute on the fly for all and cache the results)
- Work to ensure independence of the individual folds (not always easy)