Machine Learning for Networking
ML4N

Luca Vassio
Gabriele Ciravegna
Zhihao Wang
Tailai Song

The three components of ML

- Data
- Model
- Loss

ML as empirical risk minimization

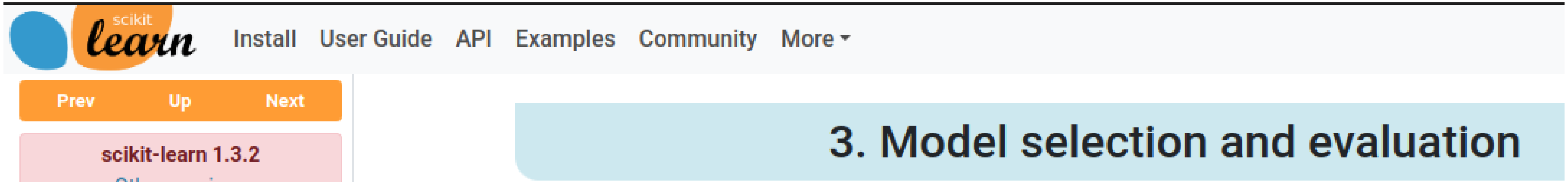
- Learn a hypothesis in model $h \in \mathcal{H}$ that incurs in **smallest empirical risk** (loss) $\widehat{L}(h|\mathcal{D})$ when predicting labels of training data points D

$$\hat{h} \in \underset{h \in \mathcal{H}}{\operatorname{argmin}} \widehat{L}(h|\mathcal{D})$$

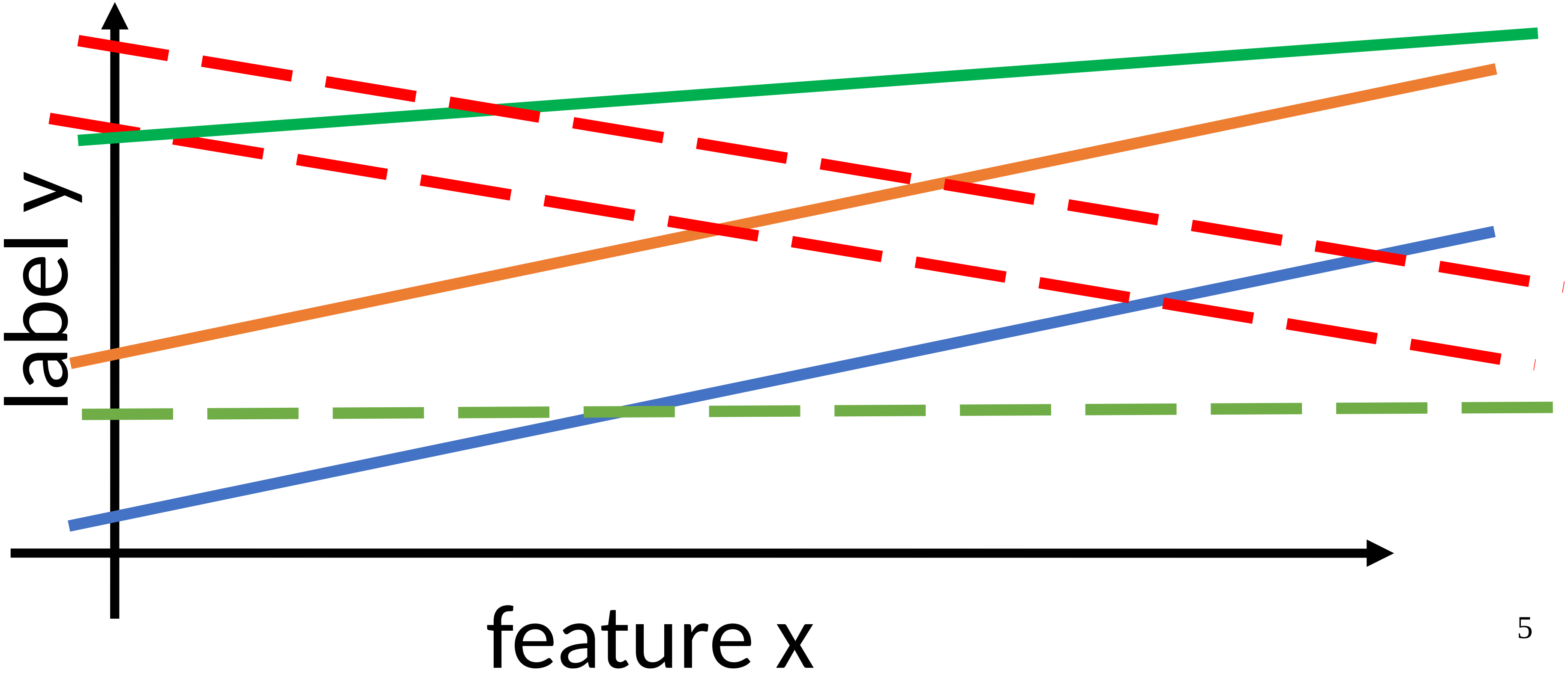
- ERM in **regressions** and **classifications**
  - Models
  - Losses

Model Selection and Validation

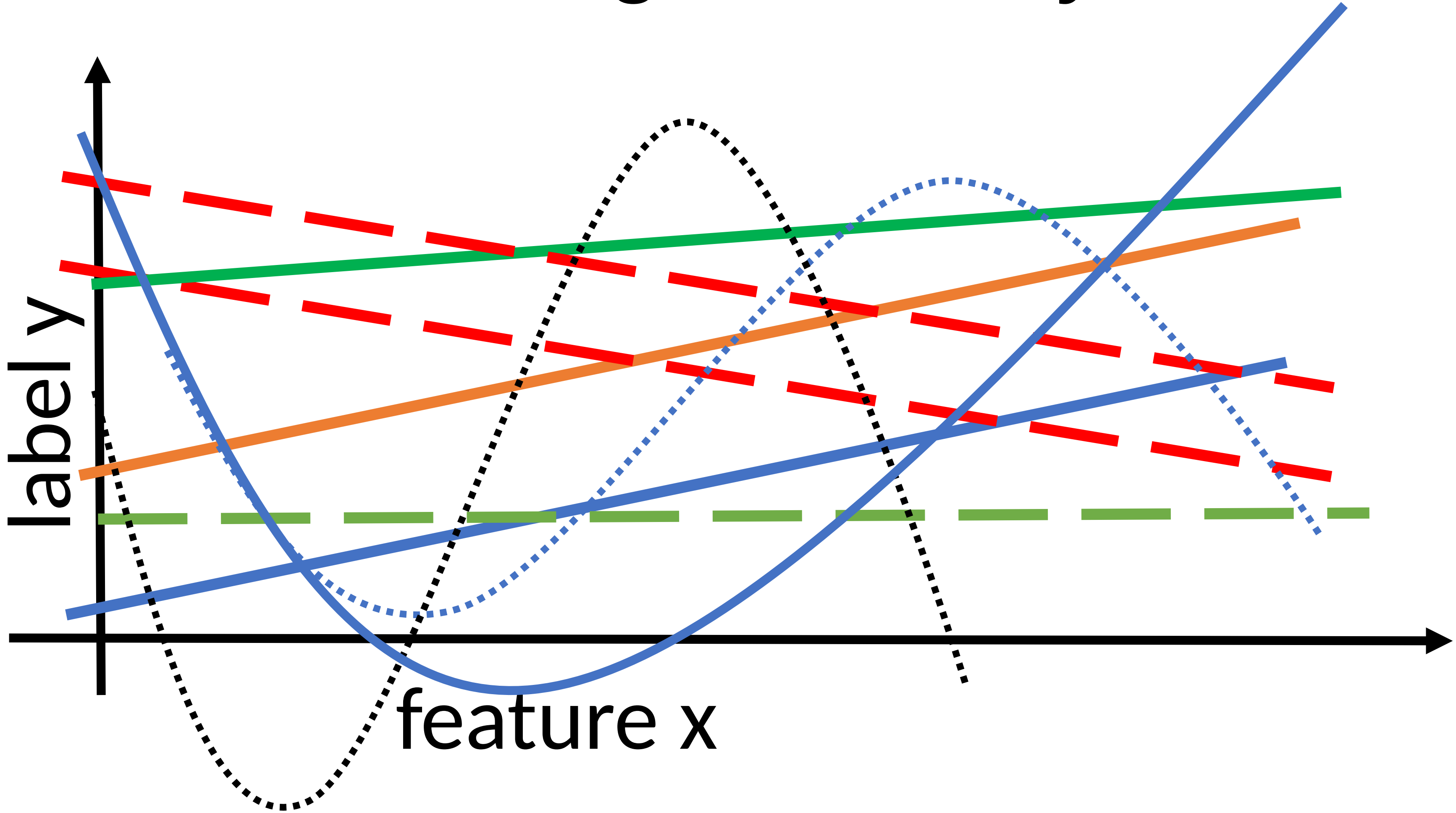
https://scikit-learn.org/stable/model_selection.html



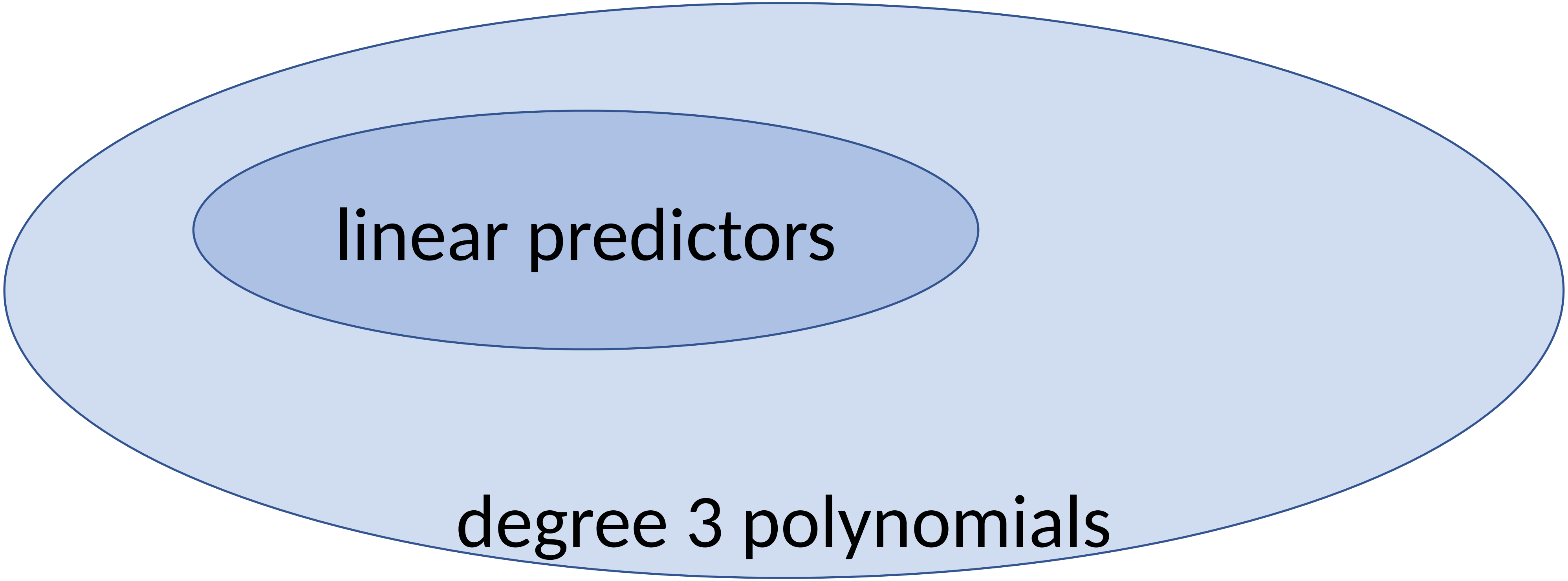
Model $H^{(1)}$= Degree 1 Polynomials



Model $H^{(3)}$= Degree 3 Polynomials



Nested models



Nested models

$$\mathcal{H}^{(n)} = \left\{ h(x) = \sum_{l=0}^{n} w_l x^l \ with\ some\ w_l \right\}$$

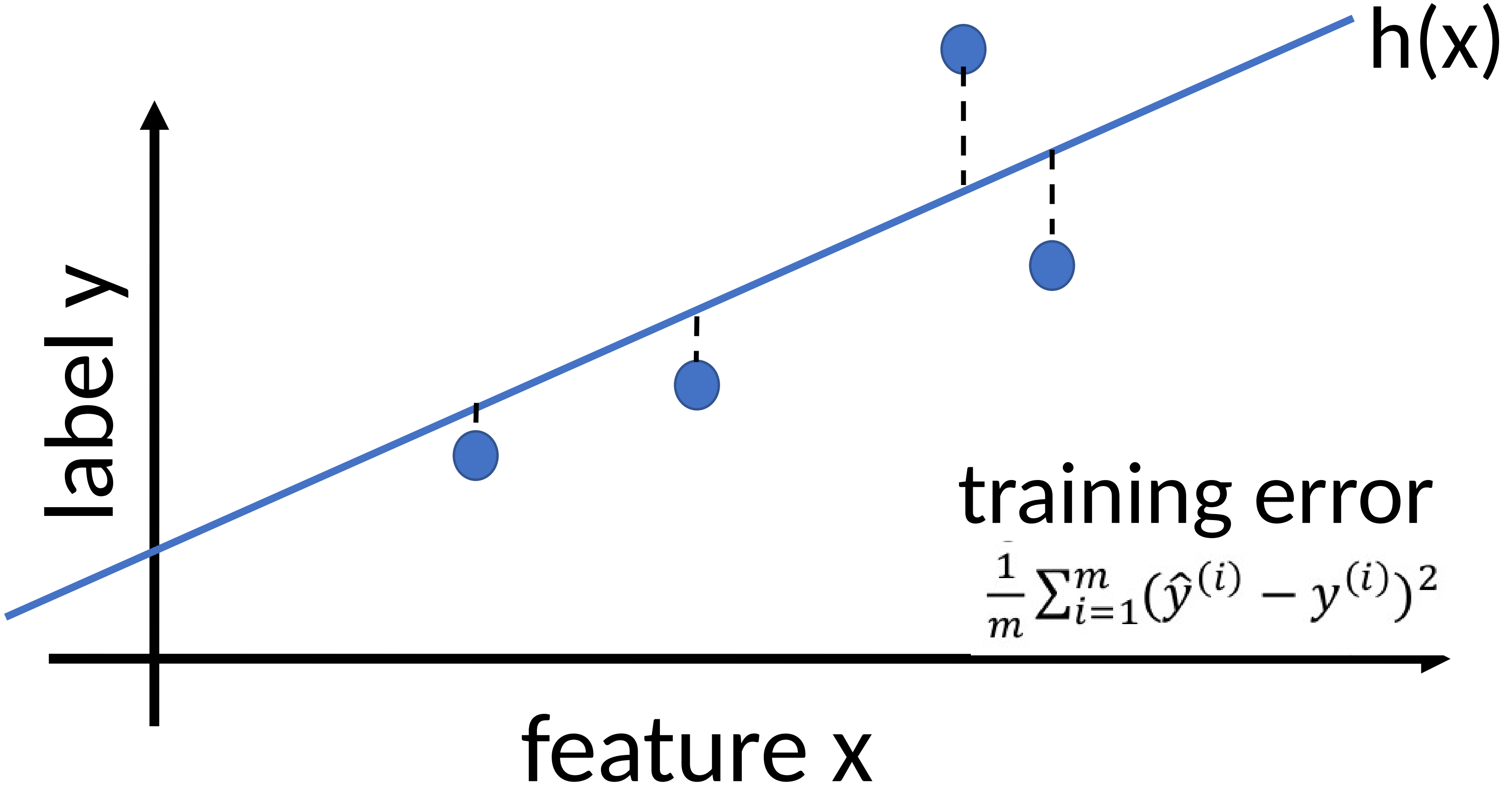
$\mathcal{H}^{(0)}$ … constant prediction (ignores feature)

$\mathcal{H}^{(1)}$ … linear hypotheses

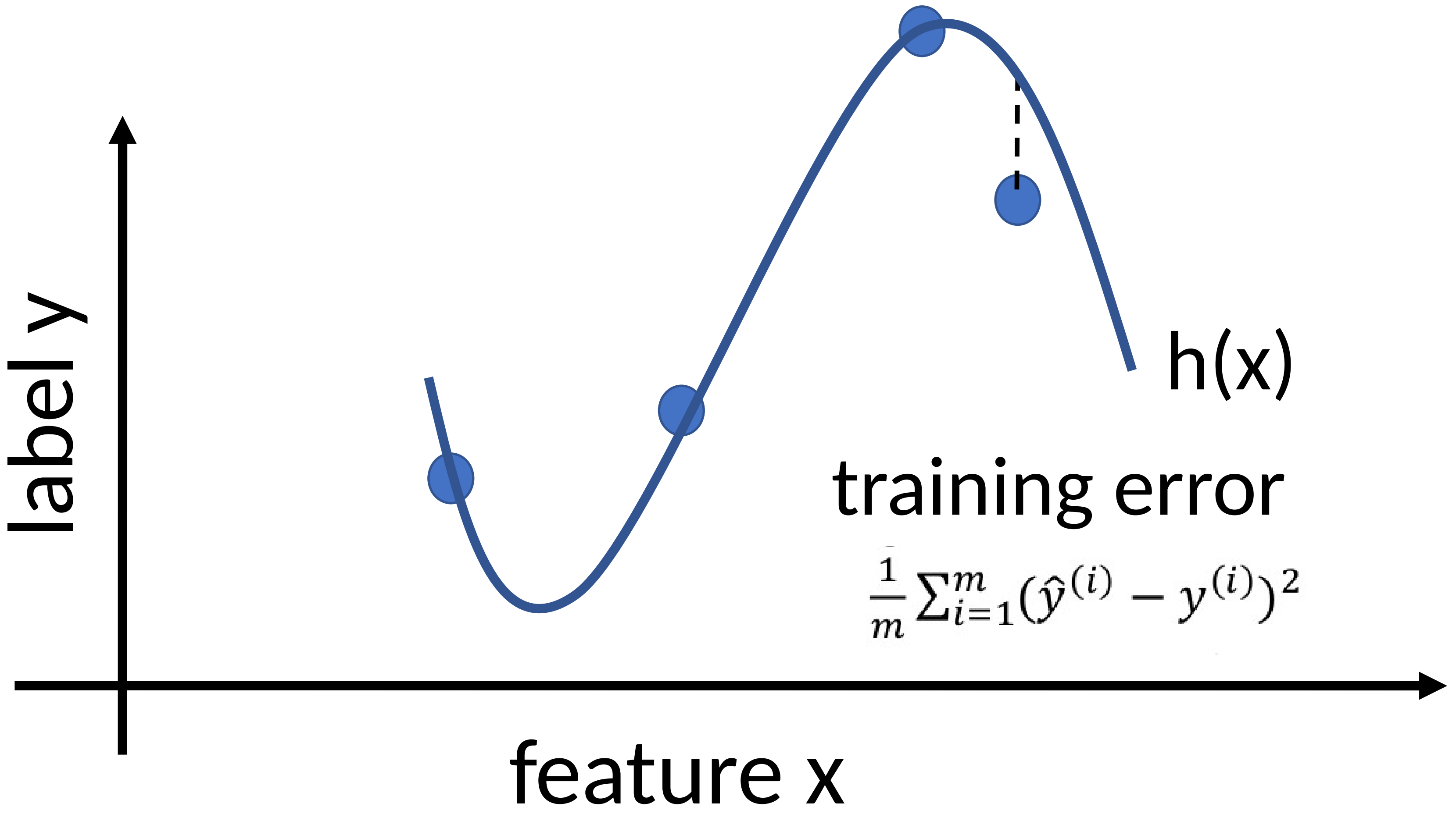
$\mathcal{H}^{(3)}$ … degree 3 polyn.

$$\mathcal{H}^{(0)} \subseteq \mathcal{H}^{(1)} \subseteq \mathcal{H}^{(2)} \subseteq \mathcal{H}^{(3)} \subseteq \ldots$$

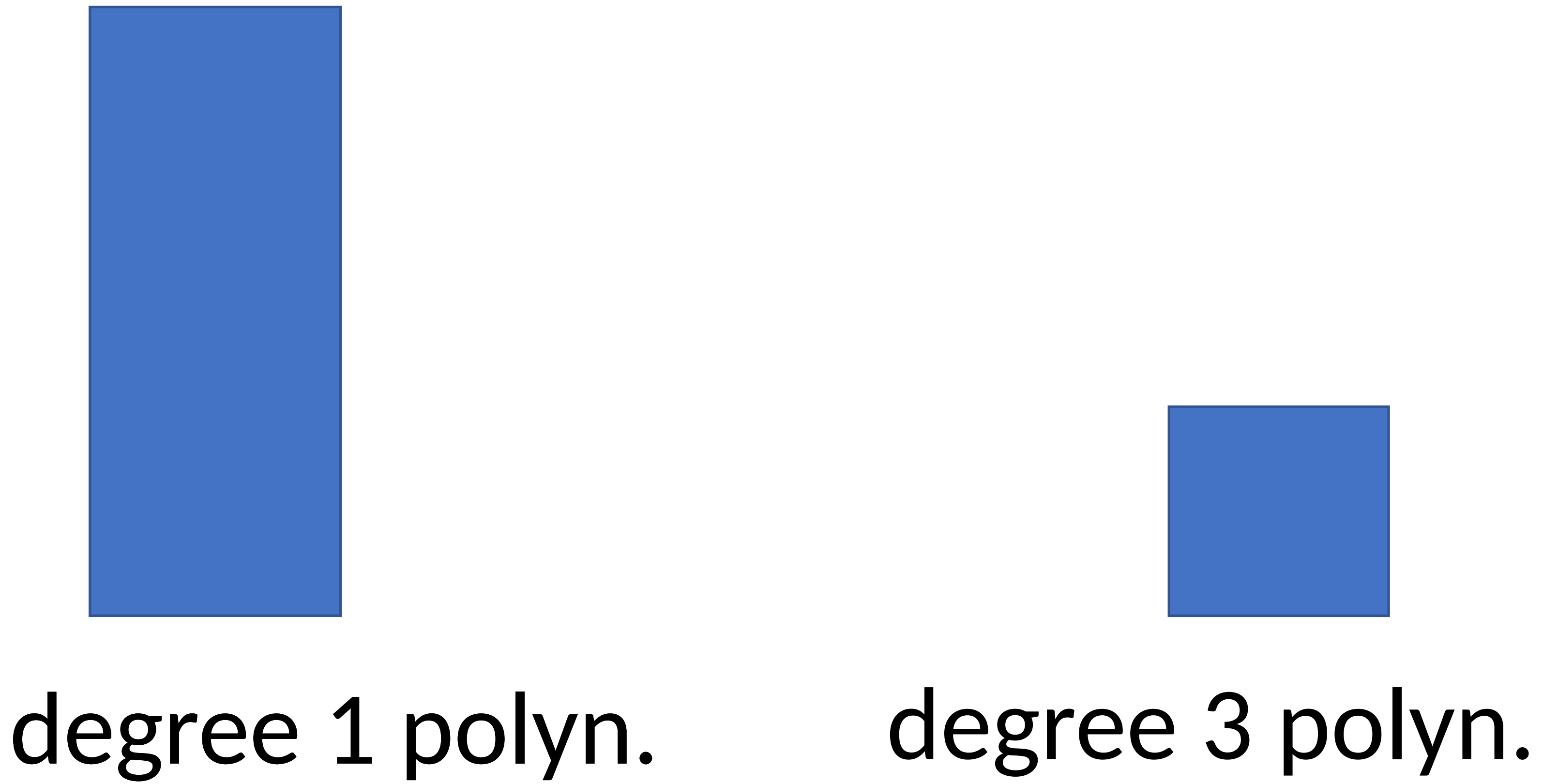
Model $H^{(1)}$= Degree 1 Polynomials



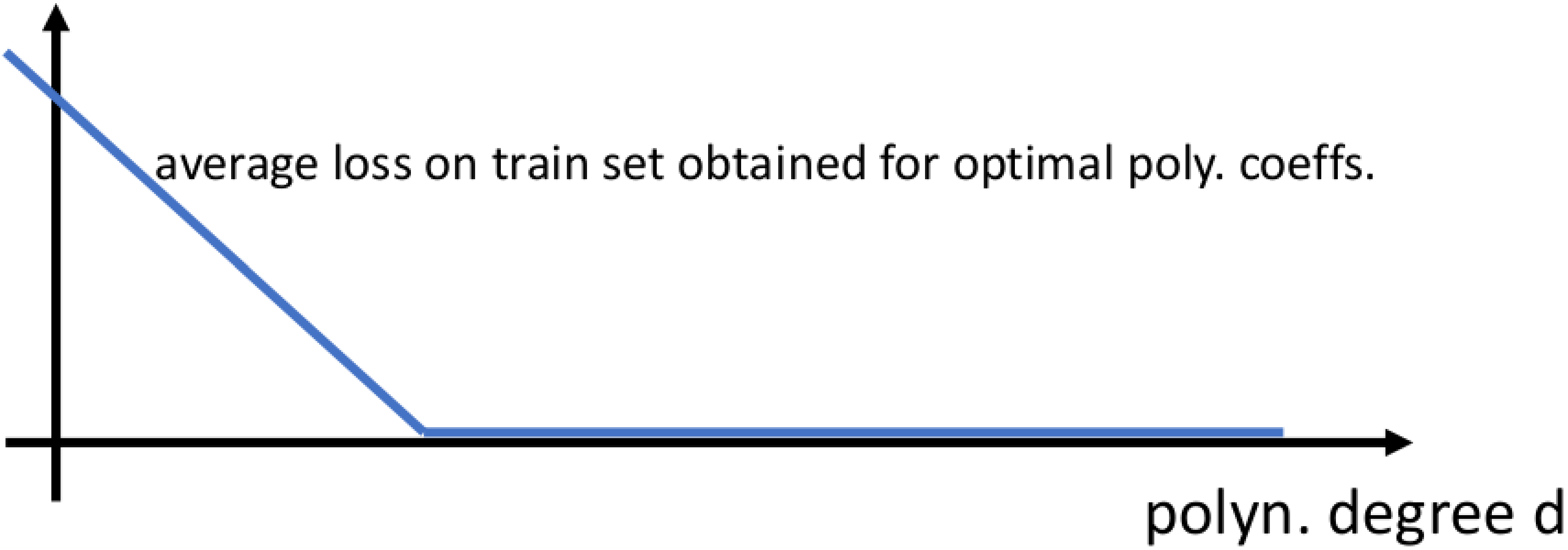
Model $H^{(3)}$= Degree 3 Polynomials



Training errors = Empirical risks



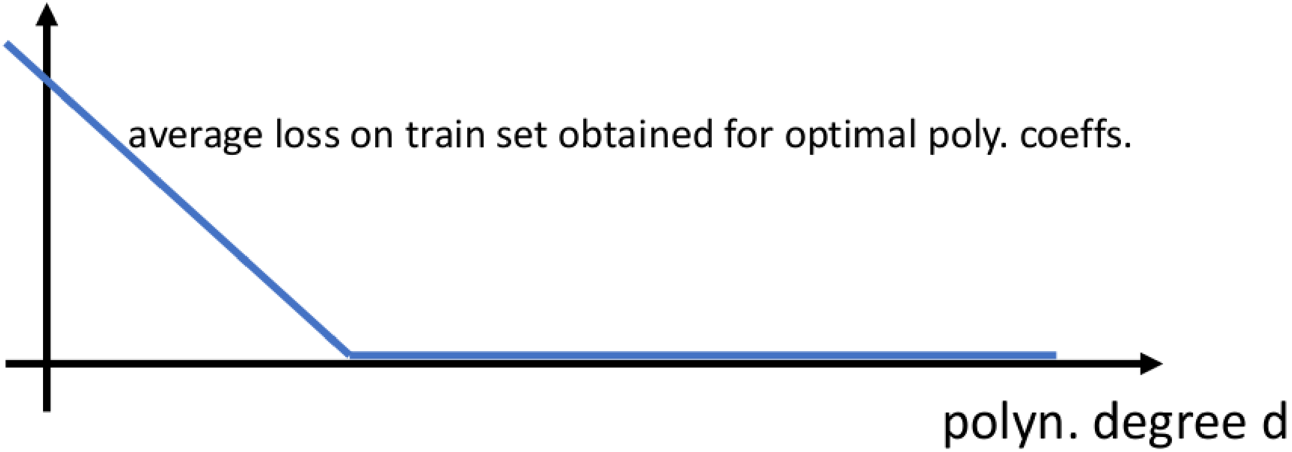
Train Error vs. Degree



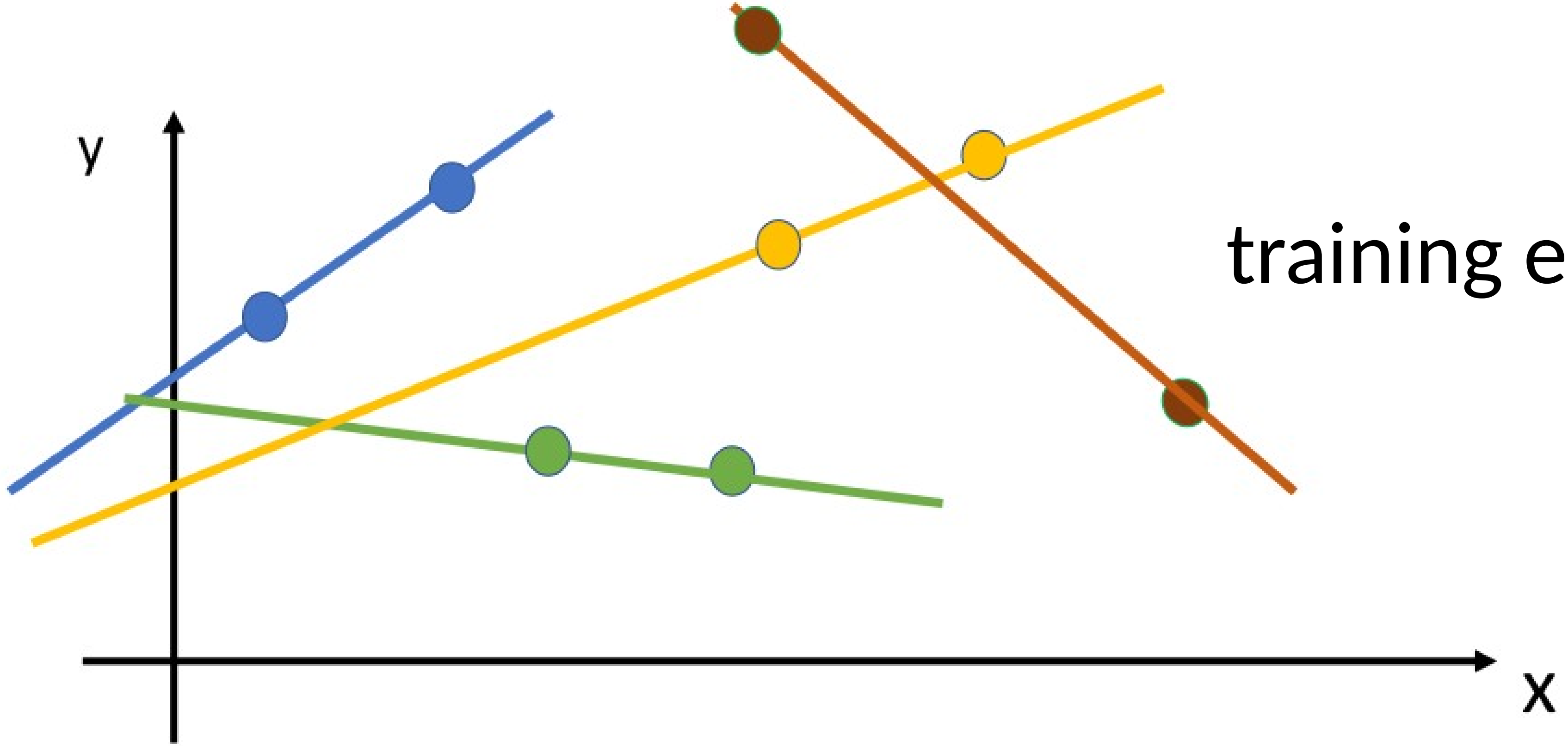
Train Error vs. Degree

We can perfectly fit (almost) any m data points using polynomials of degree n as soon as

n ≥ m-1

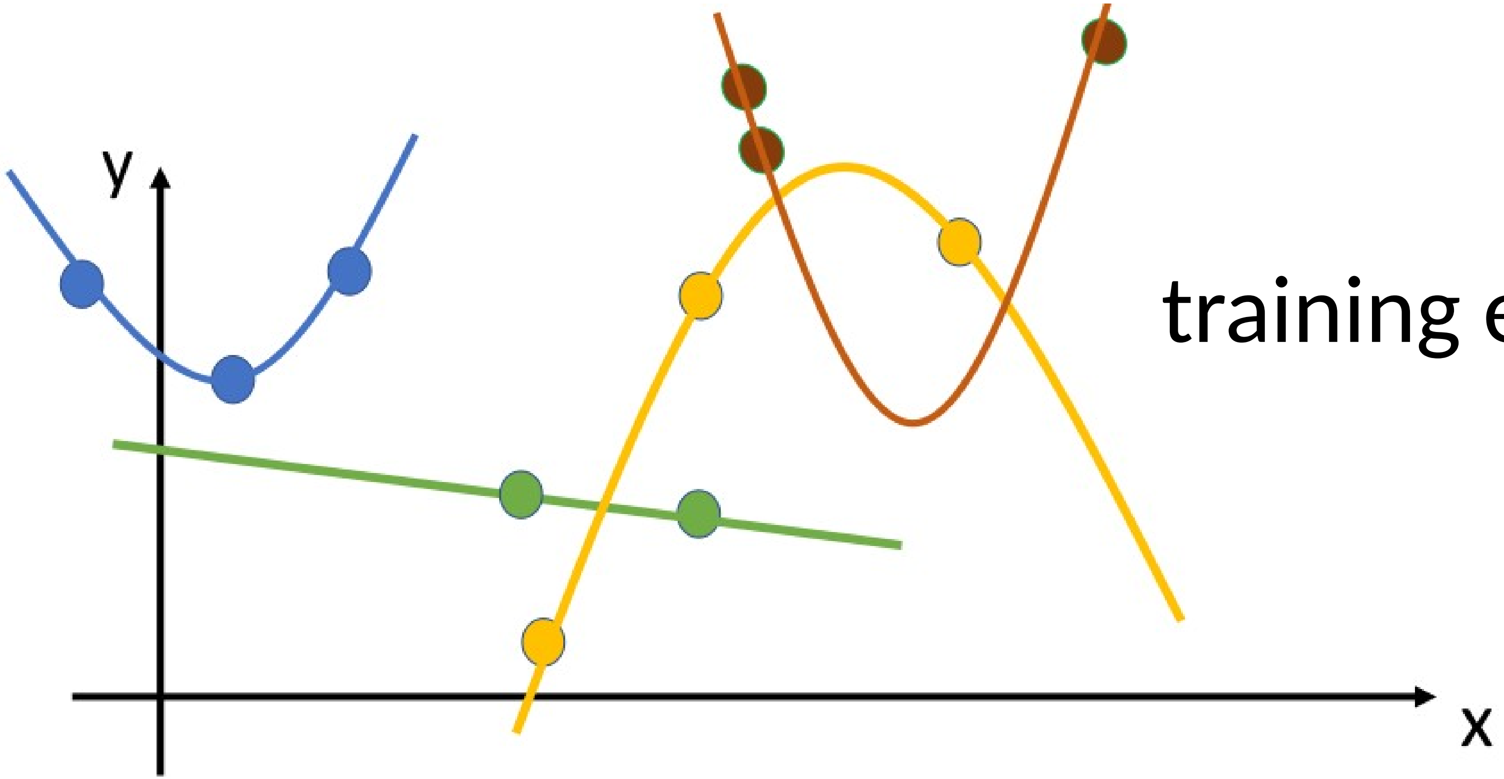


m=2 n=1



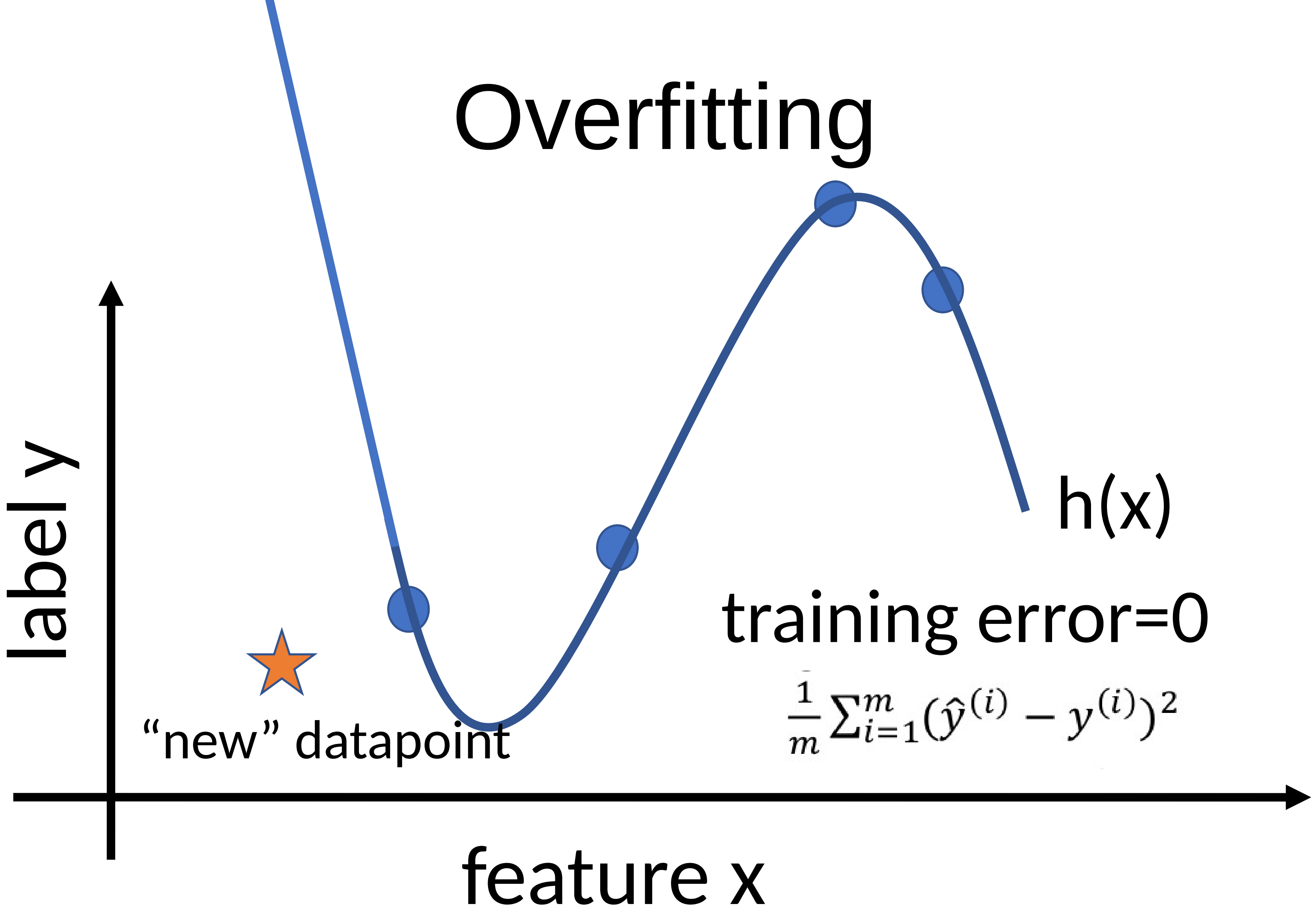
training error=0

m=3 n=2



training error=0

Overfitting



Train error is not enough

Small train error does **not guarantee** good performance **outside the training set**

outside the training set = on new/other data points

Train error is not enough

Small training error only indicates that we have **solved the ERM optimization problem**

- The model is flexible enough
- The algorithm is working well to minimize ERM

Reminder: Probabilistic Model

- data points are realizations of RVs
- joint pdf p(x,y) of features and label
- training set is a RV
- learnt hypothesis h(.) is a RV
- prediction h(x) is a RV

Why can train error mislead?

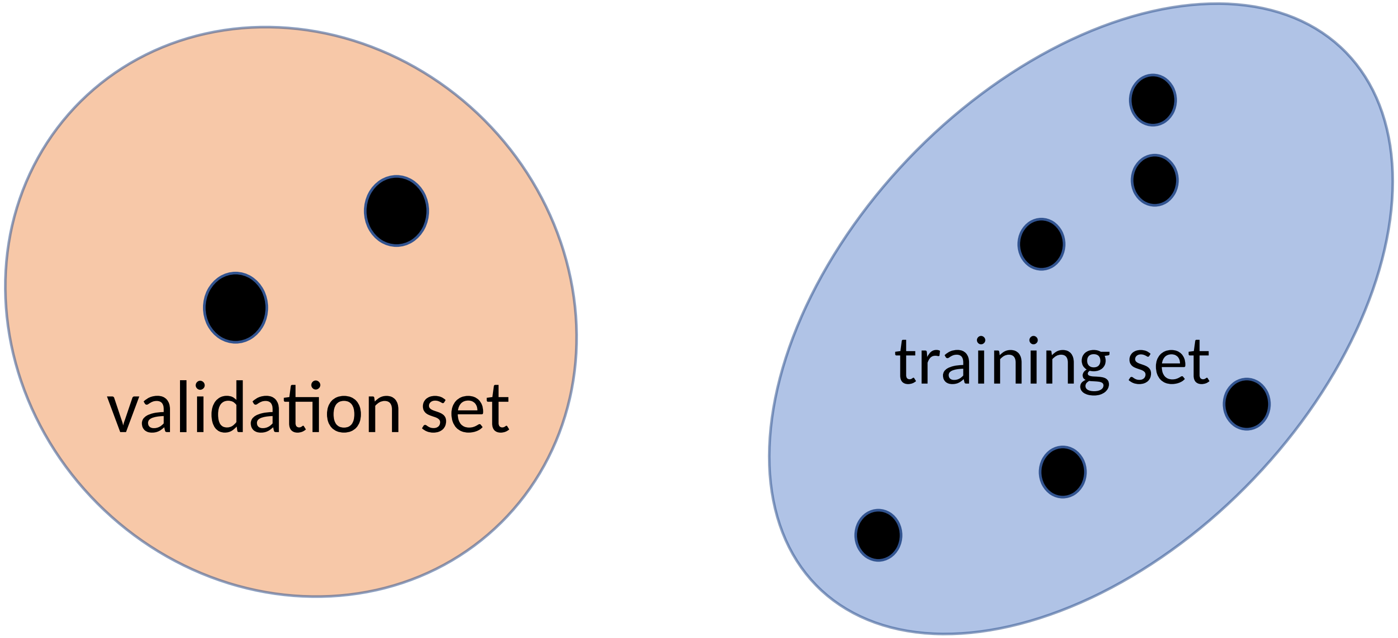
- Consider expected loss of hypothesis
- Estimate expectation using sample average
- This only works if hypothesis does not depends on data points used in average
- Does not hold for training error

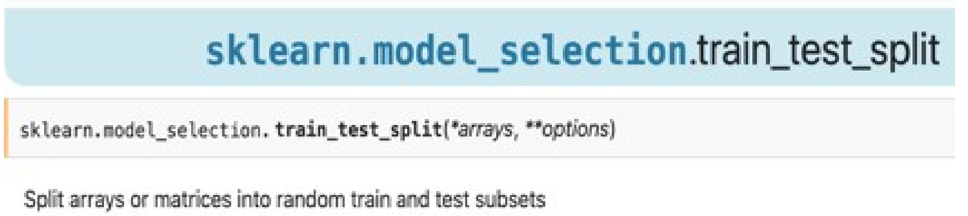
Basic Idea of Validation

Divide data points into two subsets

- **training** and **validation** set
- use training set to learn a hypothesis
- use validation set to probe outside training set (estimate expected loss)

Split Data into Training and Validation Set




sklearn.model_selection.train_test_split

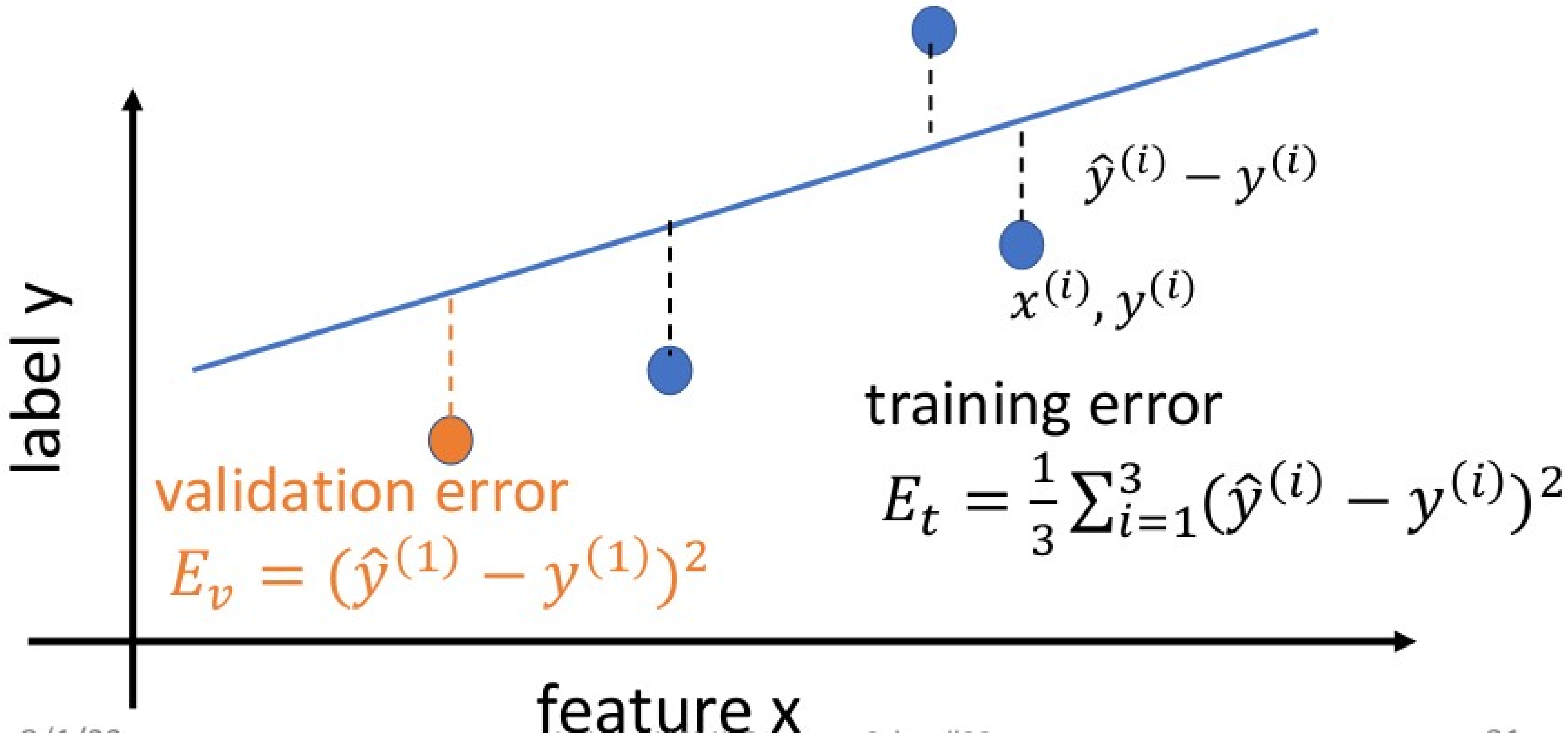
sklearn.model_selection.train_test_split(*arrays, **options)

Split arrays or matrices into random train and test subsets

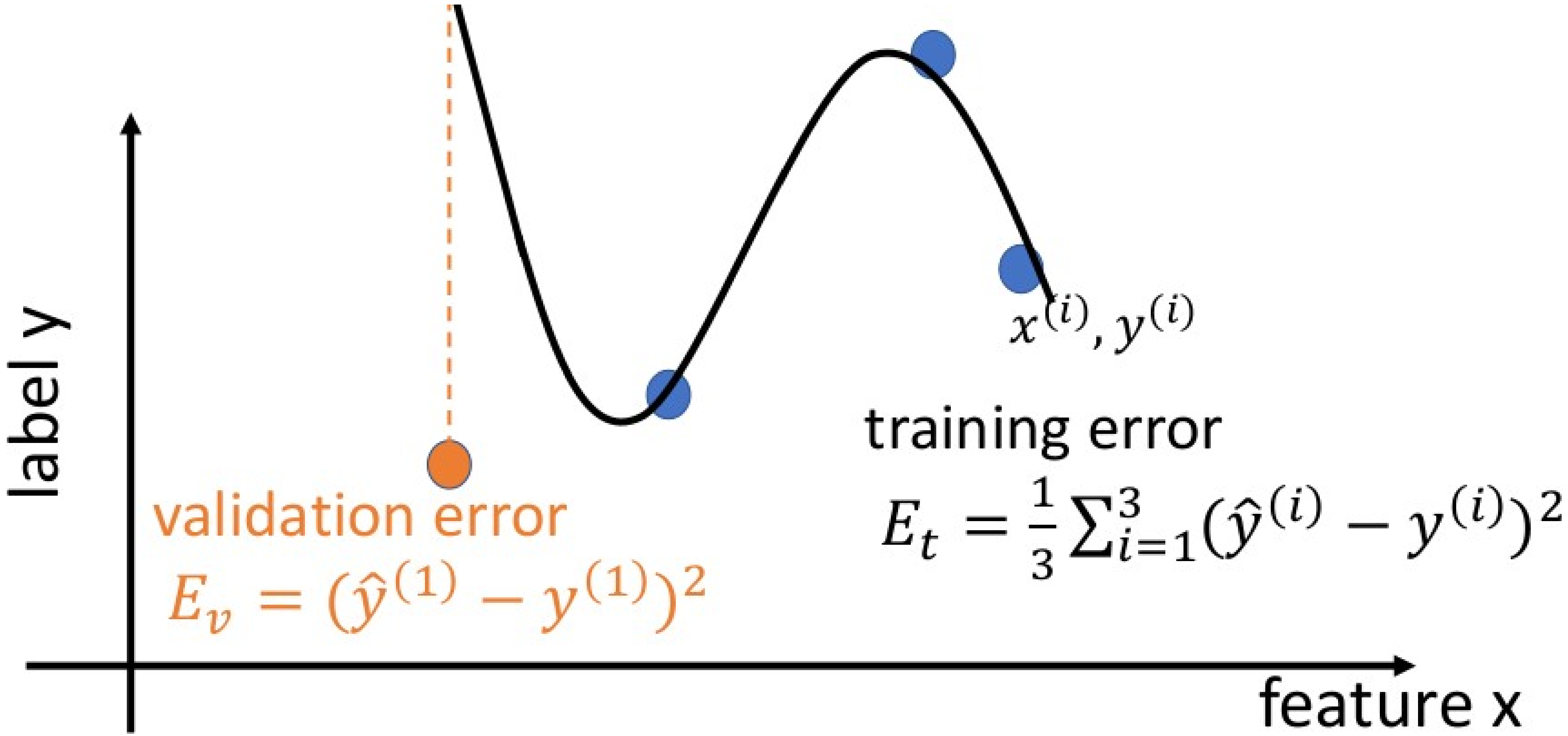
Python library:

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Train and Validation



Train and Validation



Choose model via validation error

degree 1 polyn.

degree 3 polyn.

Train/Val Error vs. Model Complexity

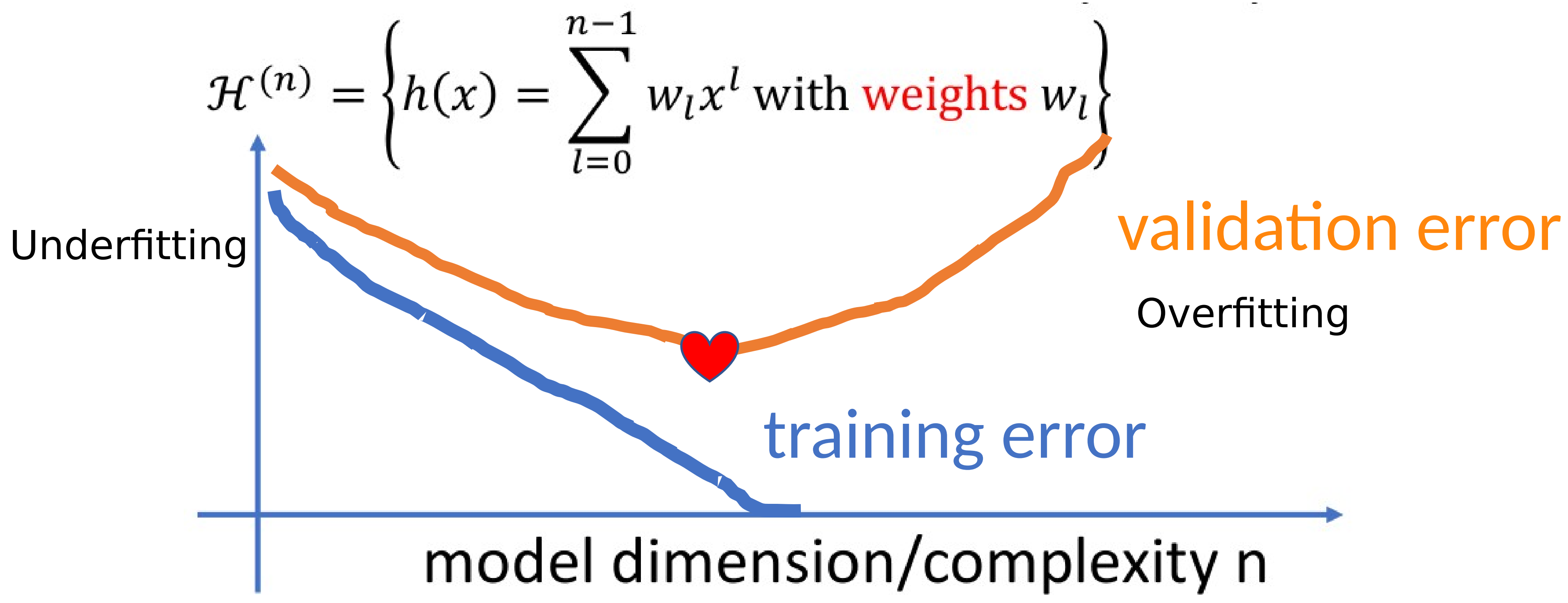
$$\mathcal{H}^{(n)} = \left\{ h(x) = \sum_{l=0}^{n-1} w_l x^l \text{ with weights } w_l \right\}$$

Train/Val Error vs. Model Complexity

$$\mathcal{H}^{(n)} = \left\{ h(x) = \sum_{l=0}^{n-1} w_l x^l \text{ with weights } w_l \right\}$$

adjust model and/or data to reach ♥

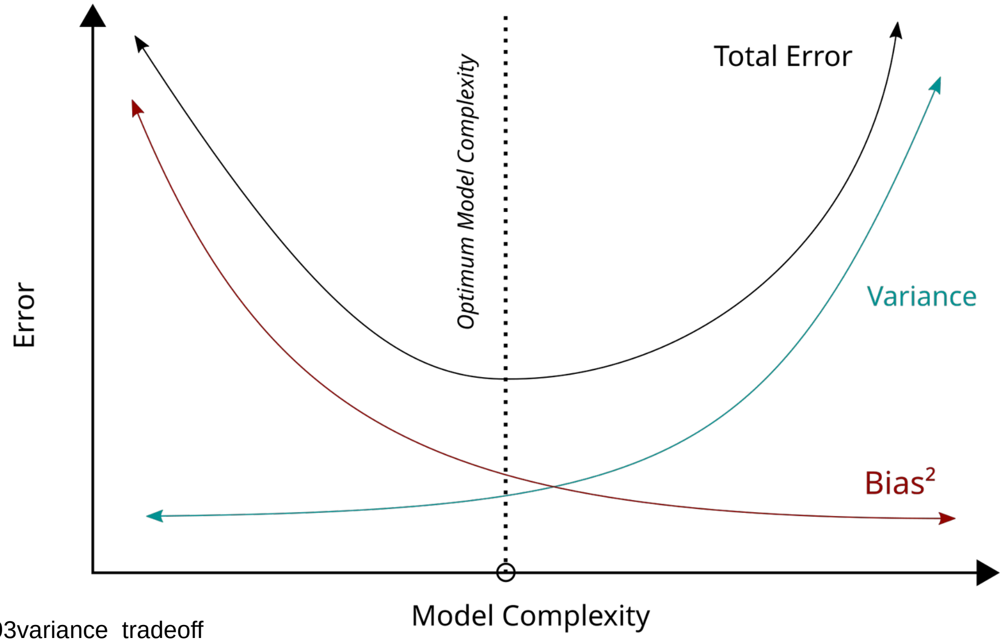
Train/Val Error vs. Model Complexity



adjust model and/or data to reach ♥

Bias vs. Variance

- Bias reflects error due to model being too small (underfitting)
- Variance reflects error due to dataset being too small/model being too large (overfitting)



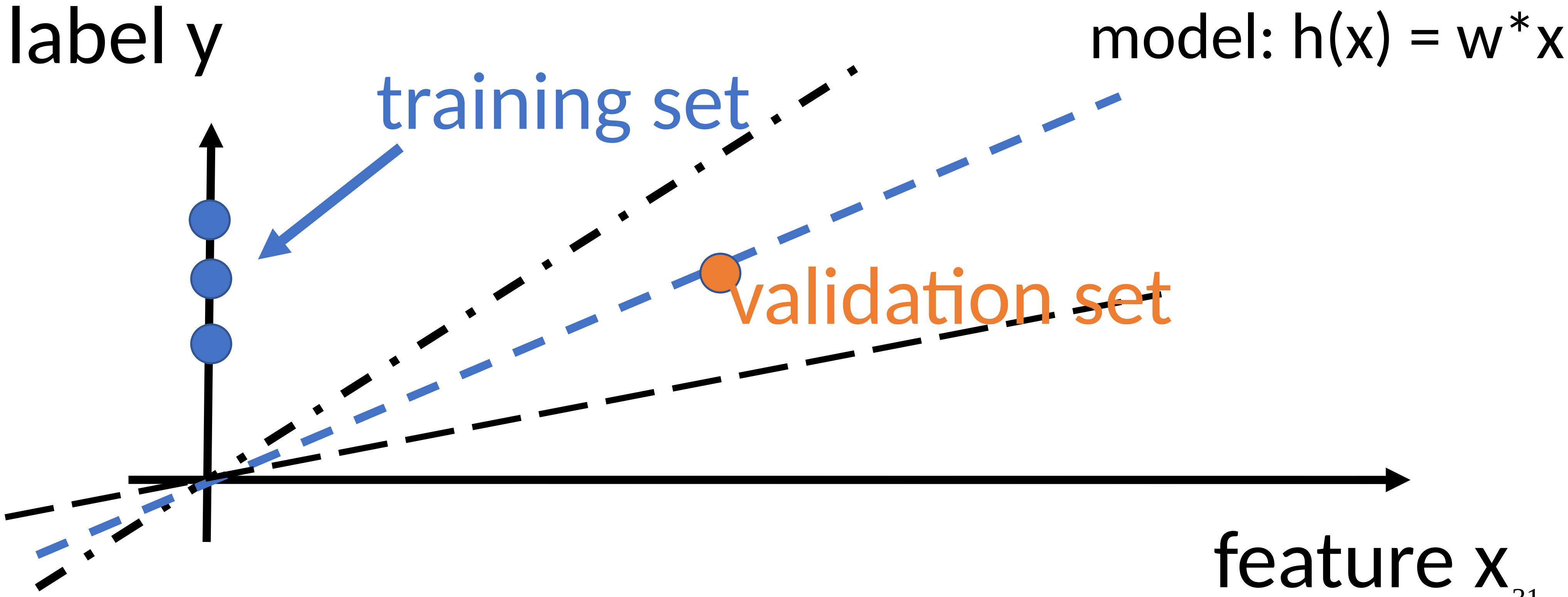
https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Train and validate in Python

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.4, random_state=42)
```

```
model.fit(X_train, y_train)
training_error = mean_squared_error(y_train,model.predict(X_train))
validation_error = mean_squared_error(y_val,model.predict(X_val))
```

Unlucky split into Train and Val set



k-fold cross validation

- might be unlucky with train/val split
- problematic for small datasets
- IDEA: randomly split several times
- “average out” unlucky splits

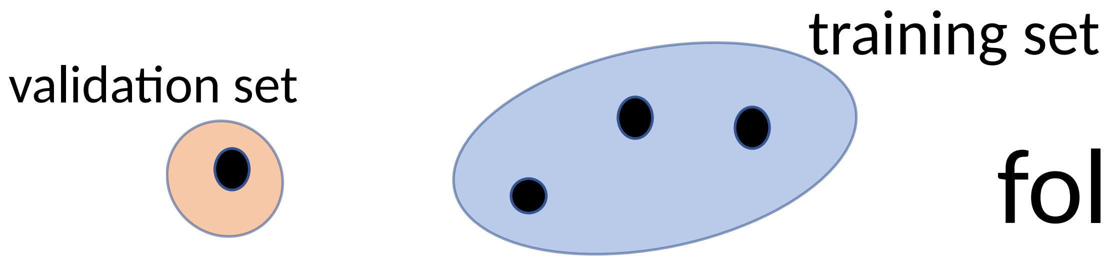
sklearn.model_selection.KFold

class `sklearn.model_selection.KFold`(*n_splits=5, *, shuffle=False, random_state=None*)

K-Folds cross-validator

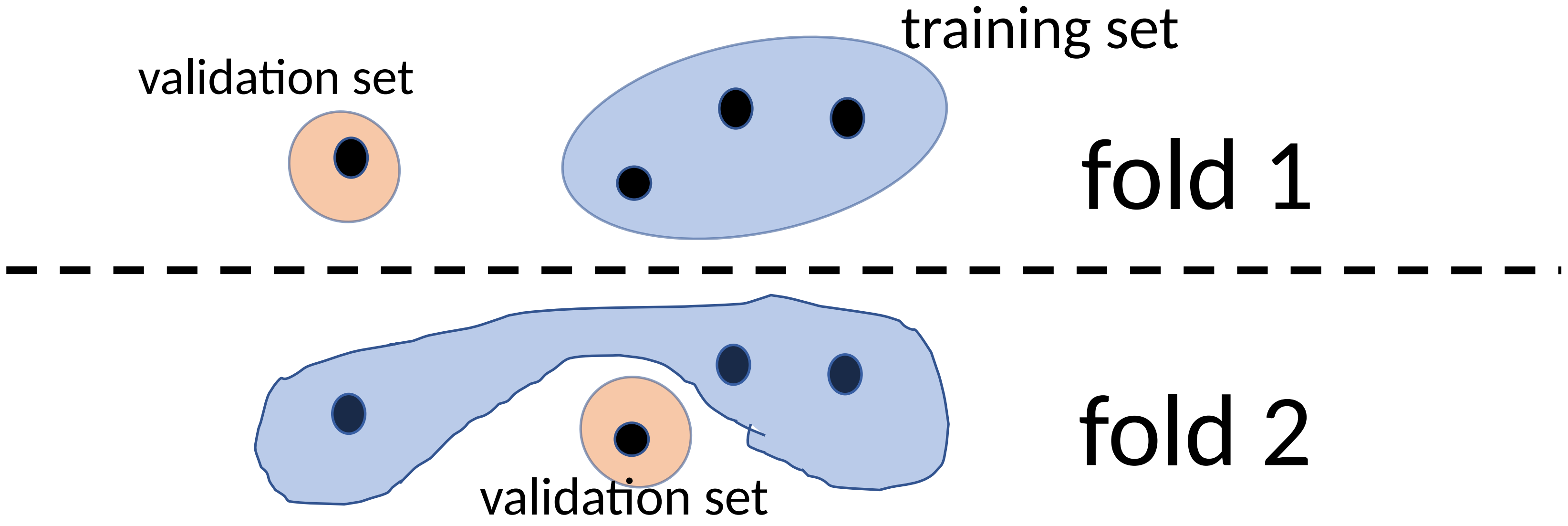
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html#sklearn.model_selection.KFold

k-fold Cross Validation

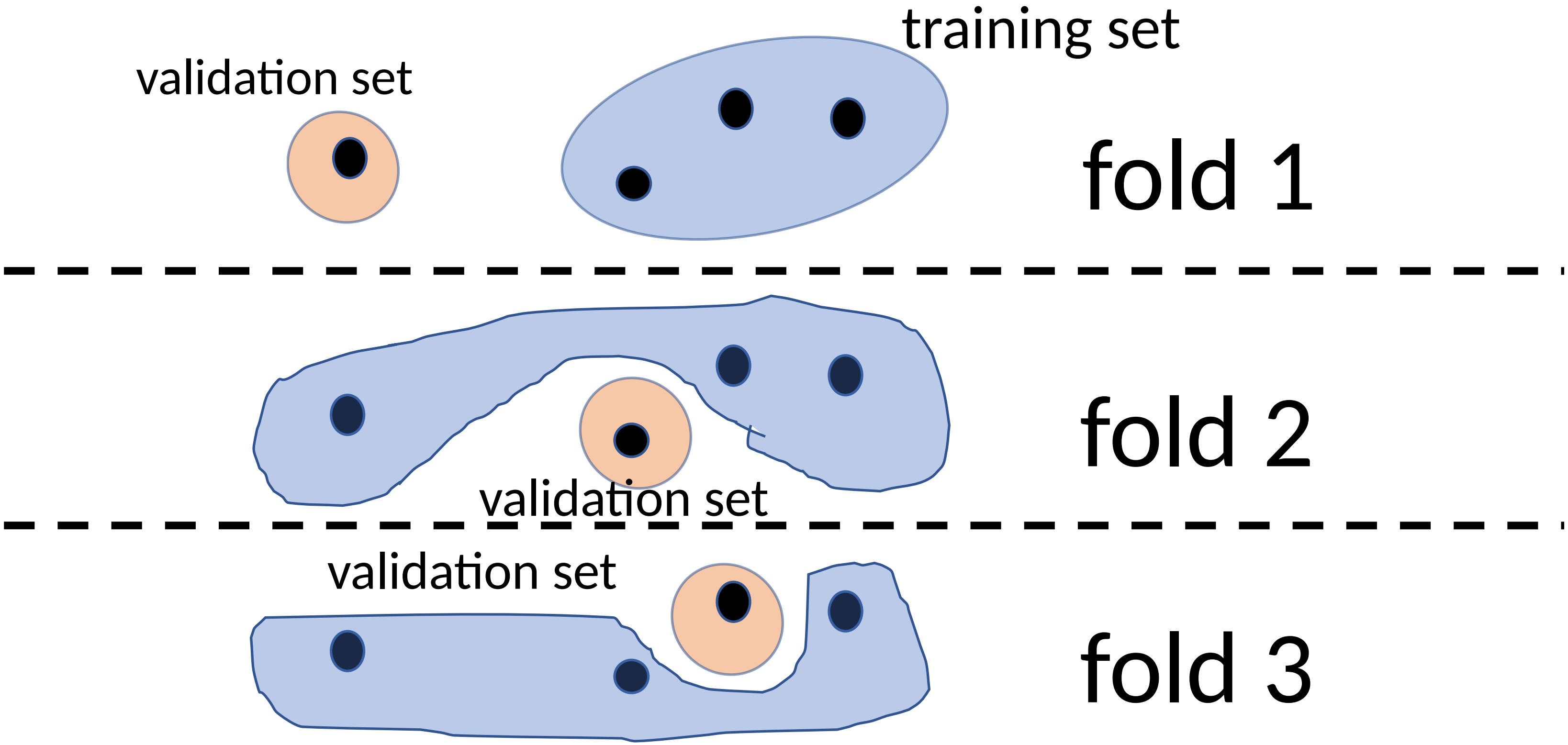


fold 1

k-fold Cross Validation



k-fold Cross Validation



k-fold Cross Validation

- Training error is the average empirical loss over the k training folds
- Validation error is the average empirical loss over the k validation folds

k-fold Cross Validation

how to choose number of folds (the “k” in k-fold CV) ?

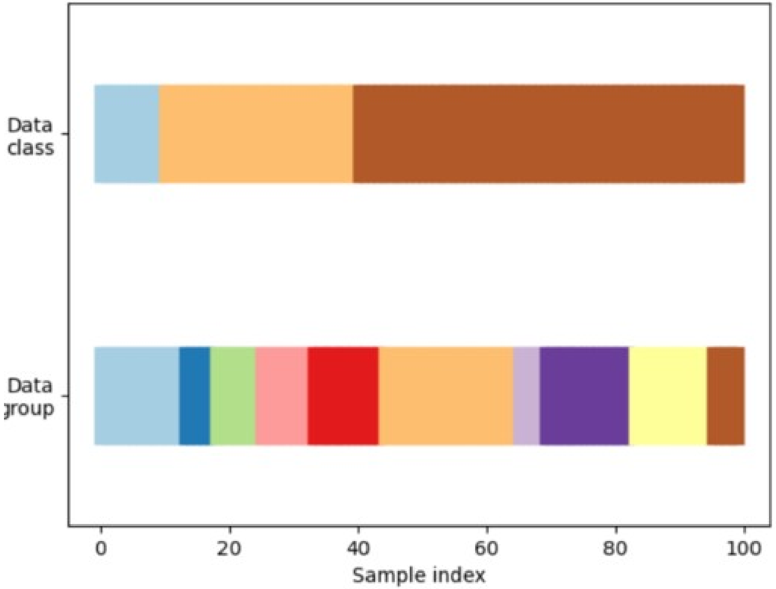
- train fold should be sufficiently large (avoid overfitting)
- validation folds should be sufficiently large (to get reliable estimate of generalization)

k-fold CV warning

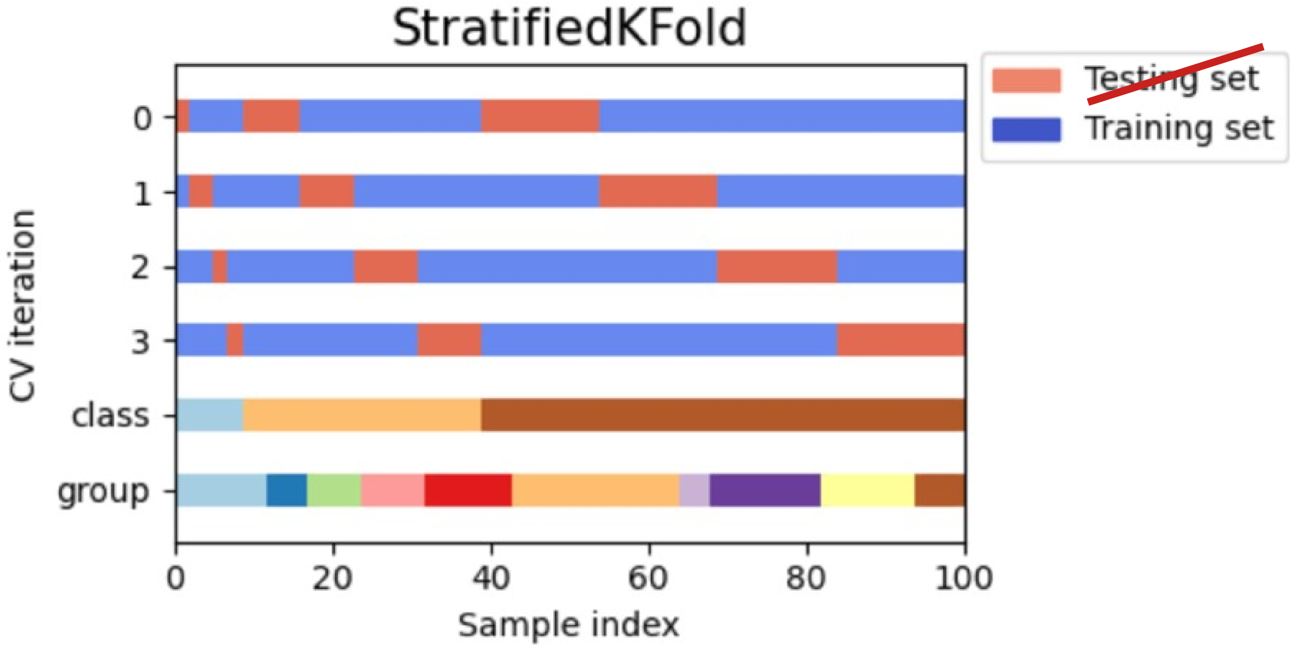
- k-fold CV requires a method to split into folds
- most basic method: evenly divide into k folds
- works if data is i.i.d. (“order of data points is arbitrary”)
- fails if data points are grouped or ordered

Imbalanced Classes and Group Structure

- data points with same label are contiguous blocks
- data points are obtained at consecutive time instants (correlations)



Class-Ratio Preserving Splitting



validation set

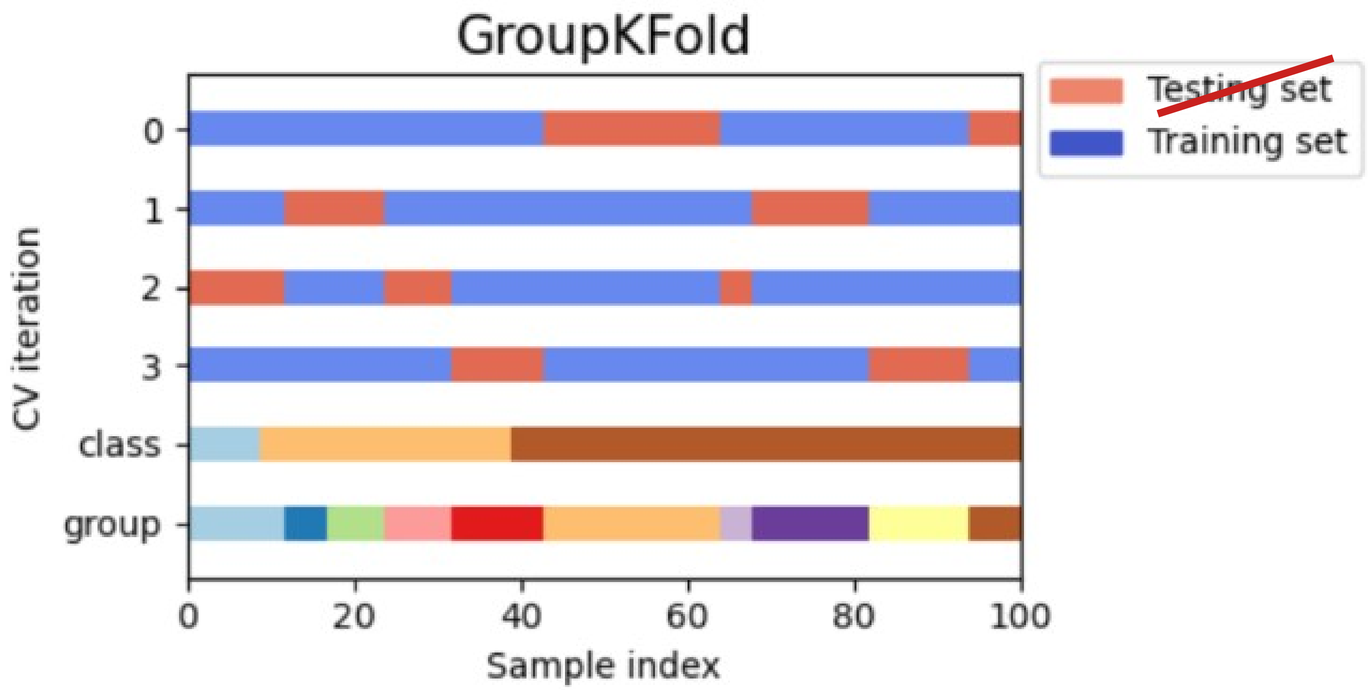
sklearn.model_selection.StratifiedKFold

class `sklearn.model_selection.StratifiedKFold`(*n_splits=5*, ***, *shuffle=False*, *random_state=None*)

Stratified K-Folds cross-validator.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

Group-Preserving Splitting



validation set

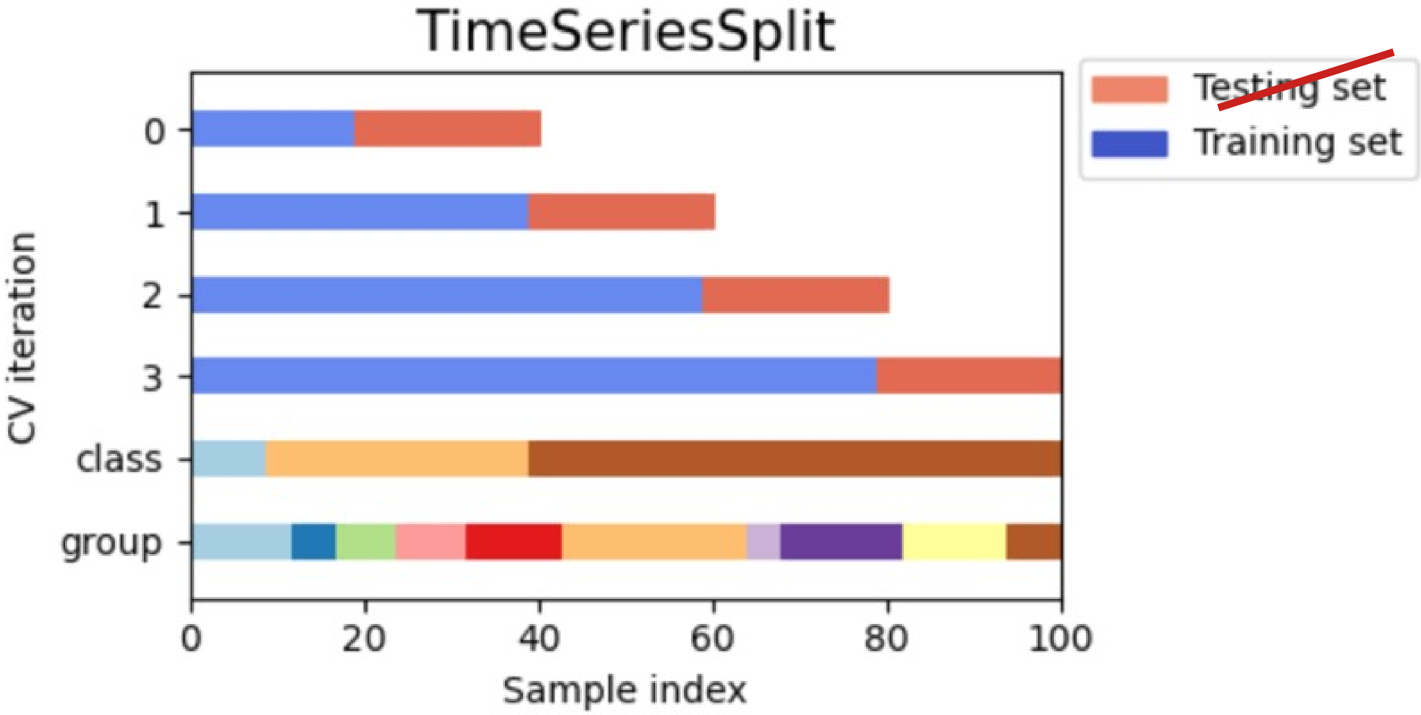
sklearn.model_selection.GroupKFold

class `sklearn.model_selection.GroupKFold`(*n_splits=5*)

K-fold iterator variant with non-overlapping groups.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GroupKFold.html

Temporal Successive Splitting



validation set

sklearn.model_selection.TimeSeriesSplit

class sklearn.model_selection.**TimeSeriesSplit**(*n_splits=5, *, max_train_size=None, test_size=None, gap=0*)

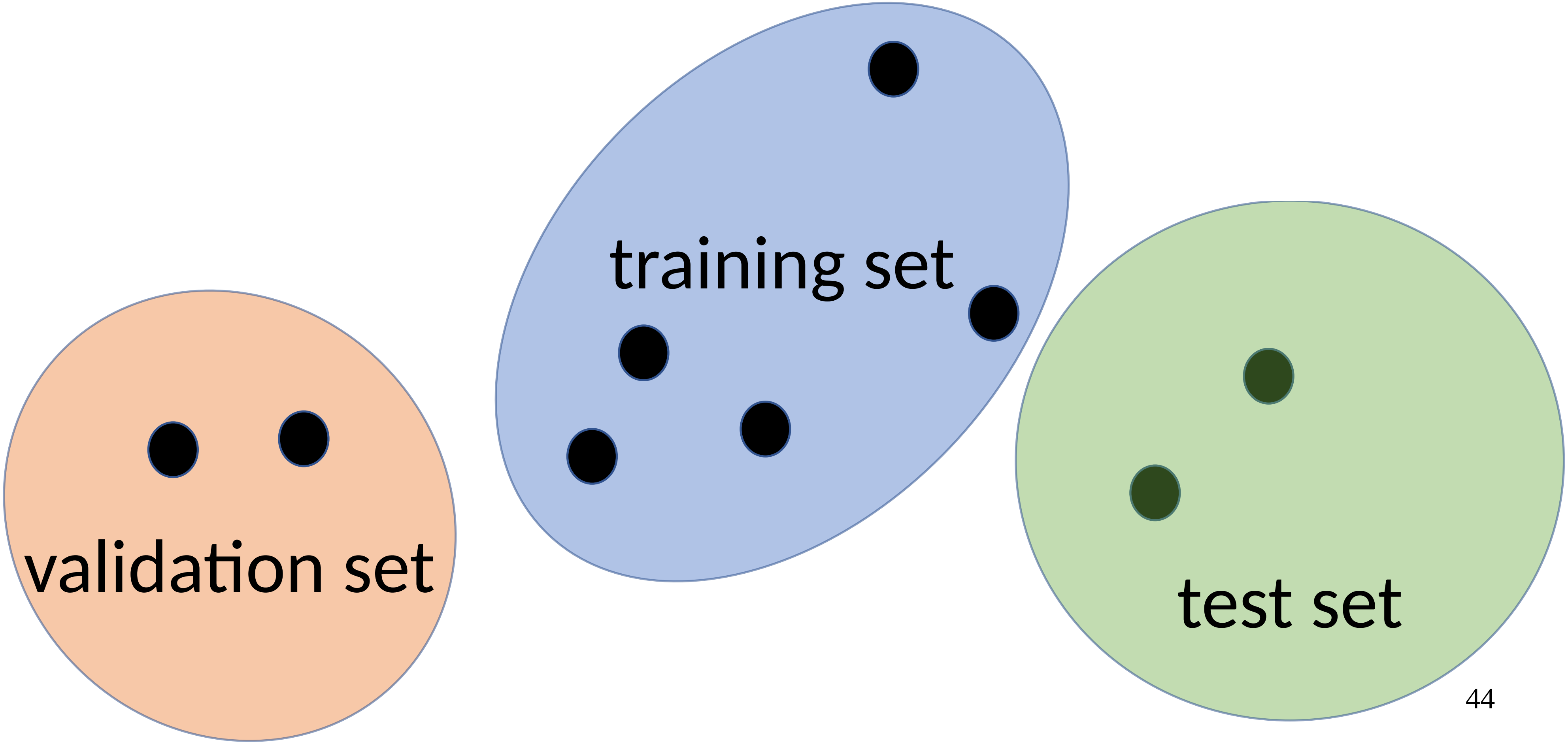
Time Series cross-validator

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

Test Set

- Chosen model with lowest validation error
- Validation error is a poor performance indicator
- Too optimistic since chosen with smallest loss
- Need a test set different from training and validation set

Split Data into Train, Val and Test Set



Model Selection Recipe

- input: list of candidate models
- for each candidate model
  - learn optimal hypothesis by minimize training error
  - compute validation error on validation set
- choose hypothesis with minimum validation error
- compute test error of on test set

Summary

- ML methods using large models tend to overfit
- probe/validate learnt hypothesis outside train set
- train on trainset, then validate on validation set
- choose model with minimum validation error
- compute test error for chosen model

Any questions?

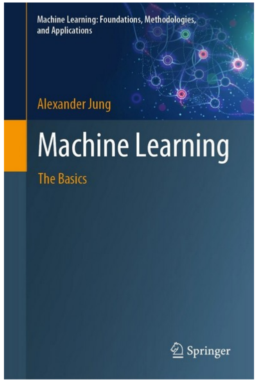
Self-assessment quiz

- Split the data into train and validation set
- Evaluate a (linear) hypothesis h on the training and on the validation by means of the squared error loss
- Repeat it by performing a 2-fold cv

Feature 1	Feature 2	Label
0	2	2
10	-25	-15
1	-2.5	-1.5
5	0	5
4	-10	-6
2	-5	-3
0	5	5
9	0	9

References: readings

- Chapter 6



Slide acknowledgments

- Alexander Jung – Aalto University