CS-C3240 - Machine Learning
Lecture 3 Mar 2021

Model Selection

Chap. 6.1 – 6.3 of mlbook.cs.aalto.fi

Alexander Jung

”Model”

=

Hypothesis Space

What I want to teach you:

- modern ML methods use large hypothesis spaces
- small training error does not rule out overfitting!
- probe learnt hypothesis on validation set
- choose the model with smallest validation error

Let’s Start with a Course Recap

What are three main components of machine learning?

1. Data

Data

- set of “data points” (atomic unit of information)
- each data point has features and labels
- features =properties that can be measured easily
- labels = higher-level facts or quantities of interest

Data Point = ”Some Ski Day”

features x : pixel RGB values of webcam snapshot

label y : maximum daytime temperature

Data = Bunch of Data Points

feature vector $\boldsymbol{x}^{(1)}$
label $y^{(1)}$

$\boldsymbol{x}^{(5)}, y^{(5)}$

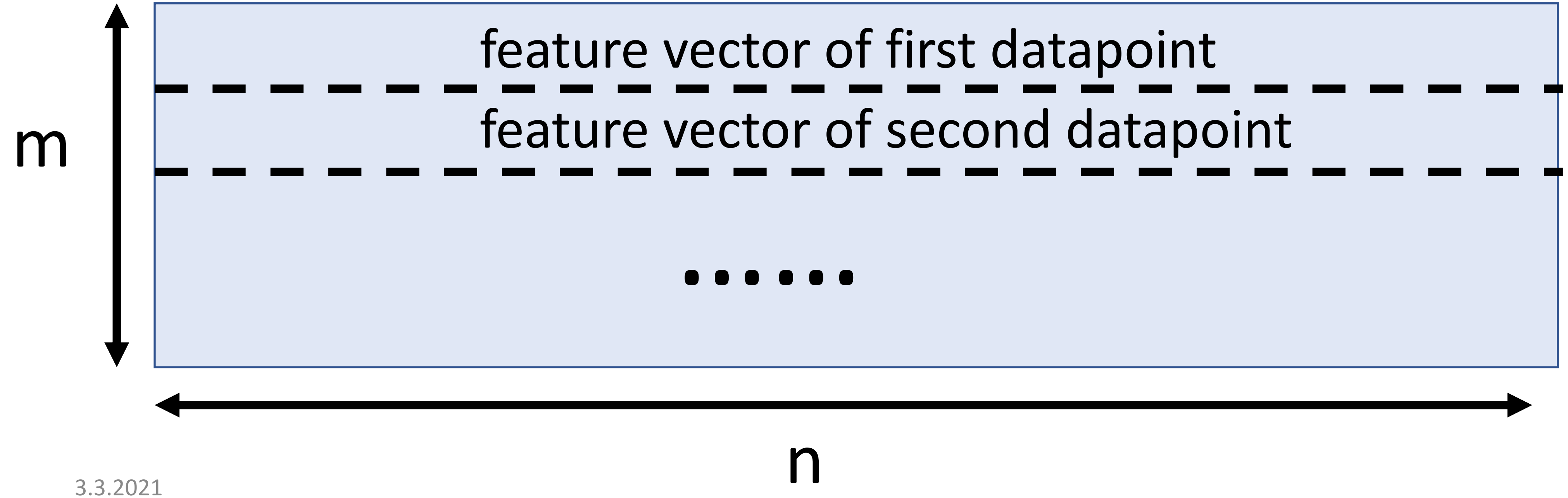
$\boldsymbol{x}^{(3)}, y^{(3)}$

$\boldsymbol{x}^{(2)}, y^{(2)}$

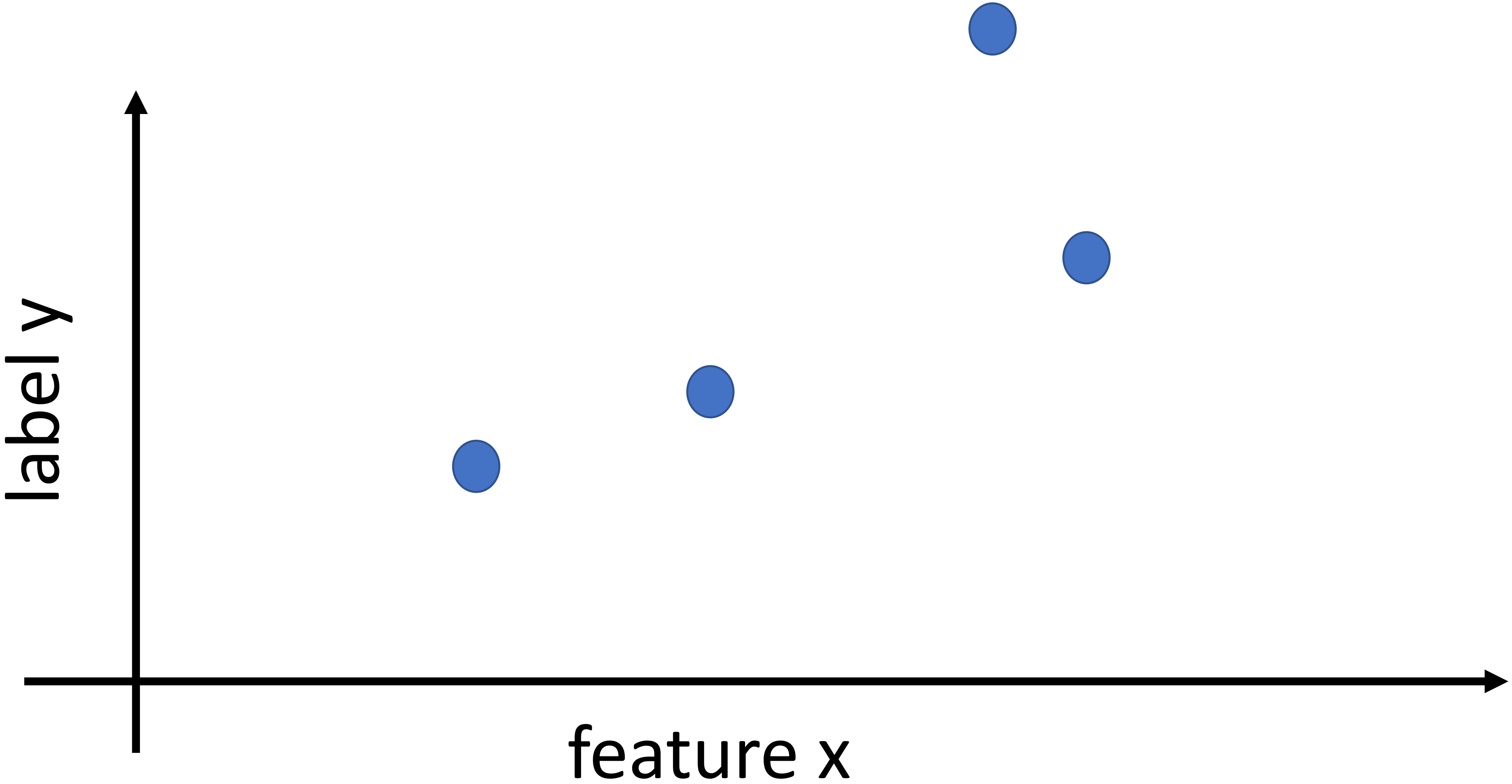
$\boldsymbol{x}^{(4)}, y^{(4)}$

Key Parameters of Datasets

- number m of data points
- number n of features



Scatterplot – First Look at Data



2. Hypothesis Space/Model

How Many Hypotheses Are There?

predict

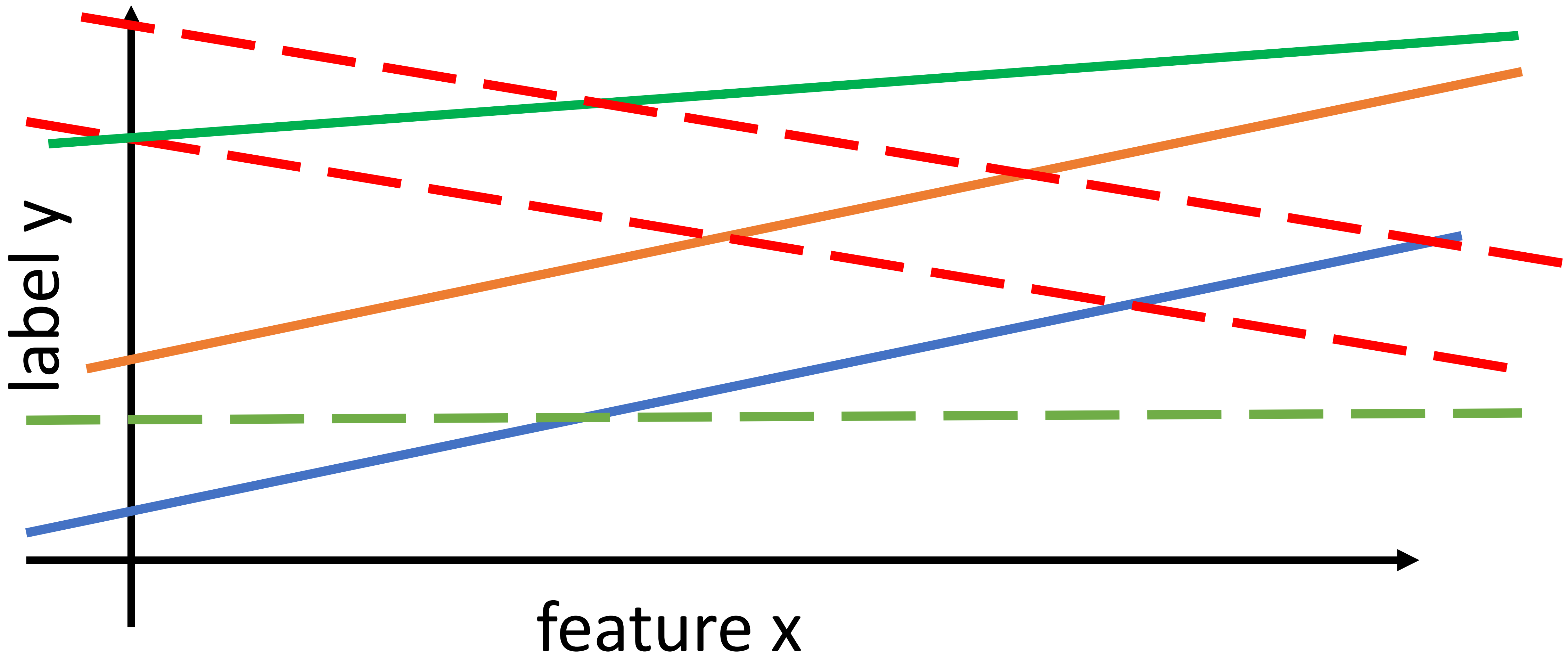
h(x)

predicted max temp.

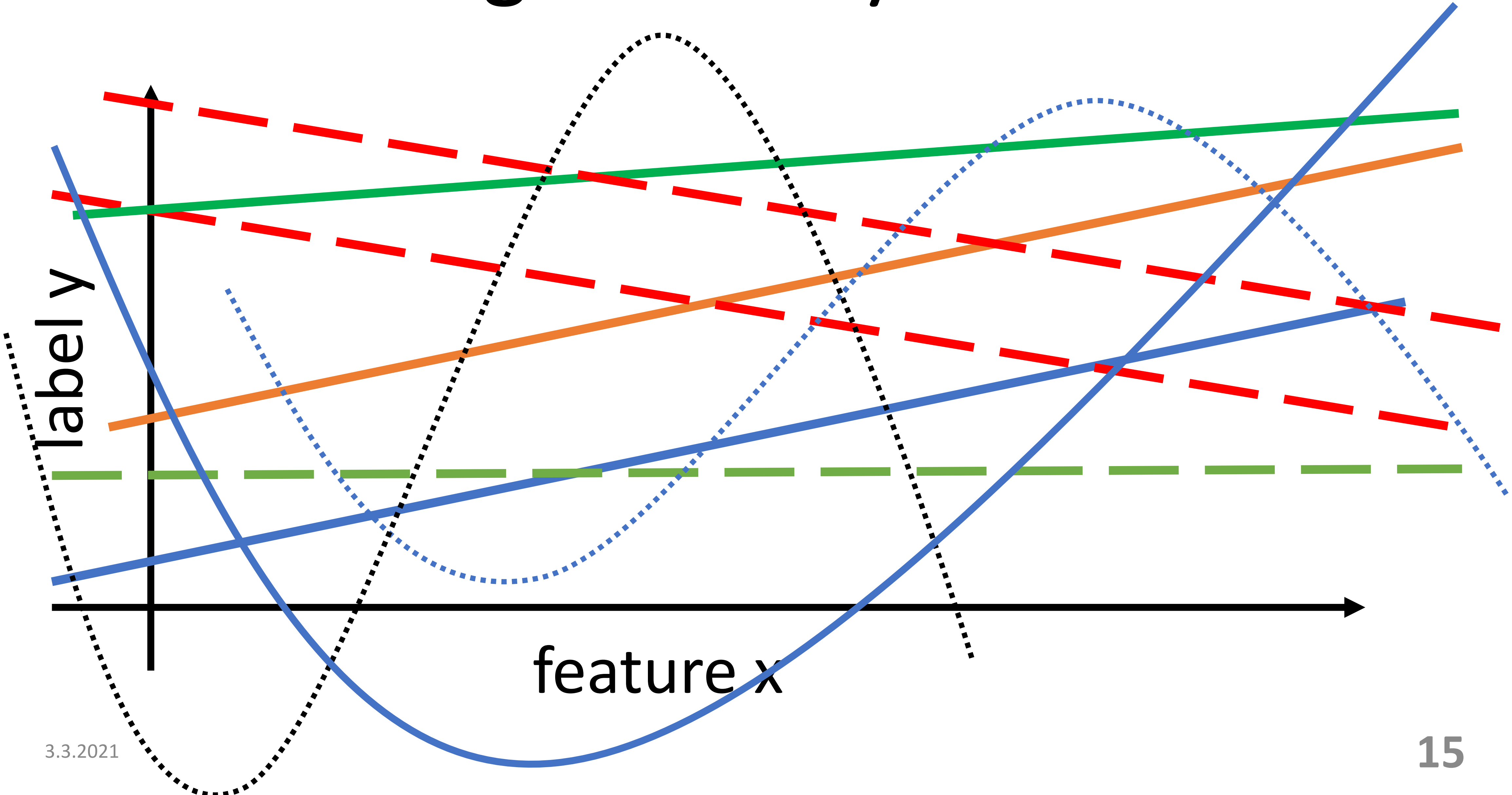
$$\hat{y} = h(x) = -2.3$$

feature x = -10

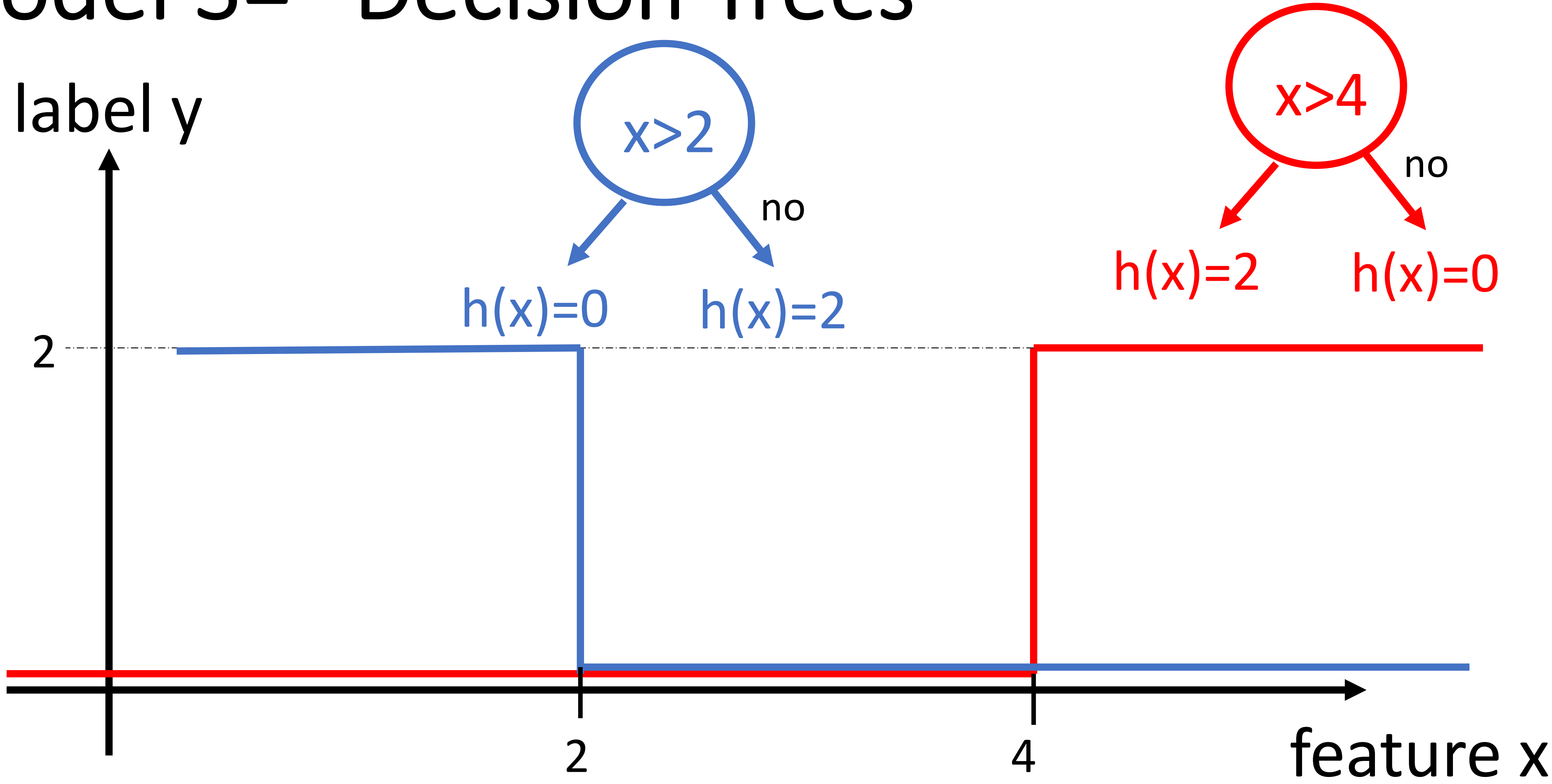
Model 1="Linear Predictors/Degree 1 Polyn."



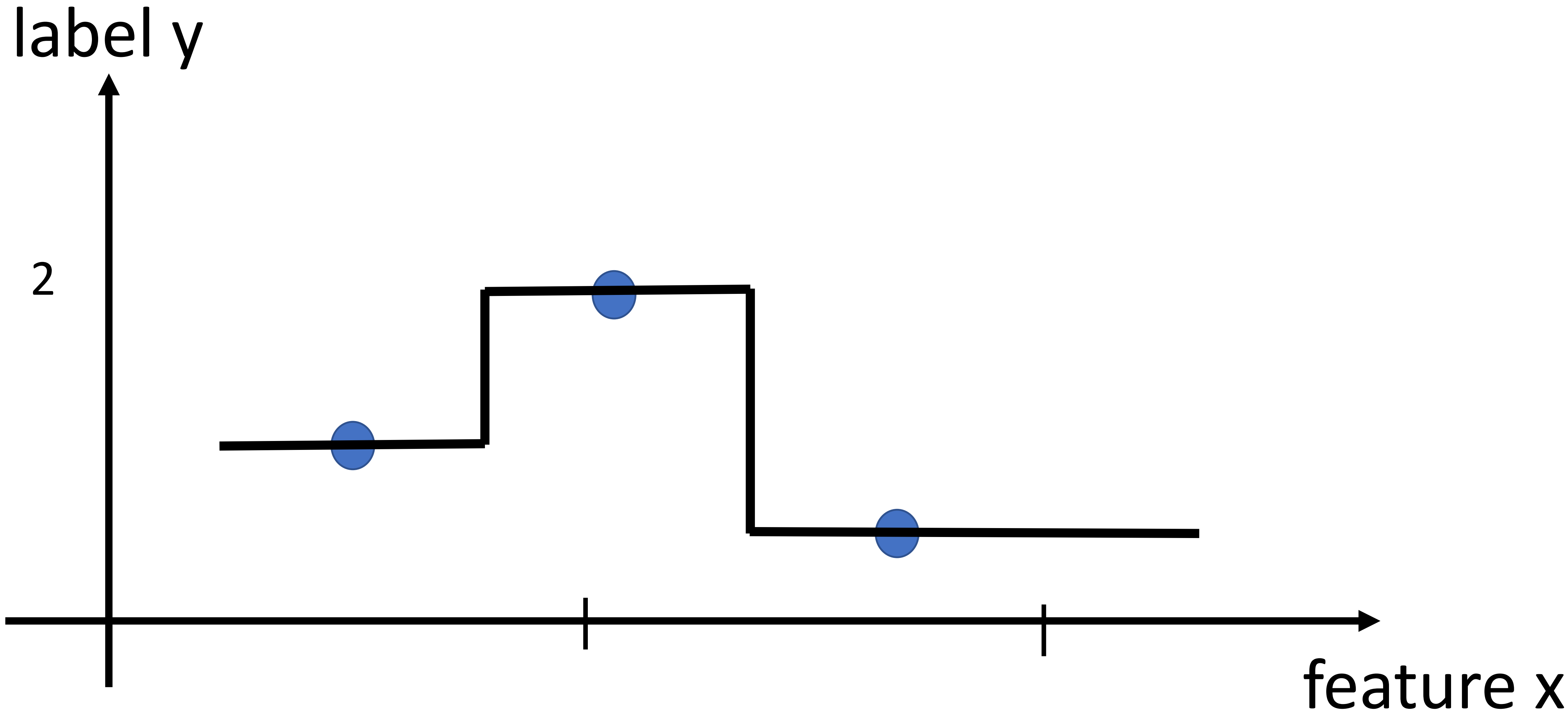
Model 2= “Degree 3 Polyn. Predictors”



Model 3= “Decision Trees”



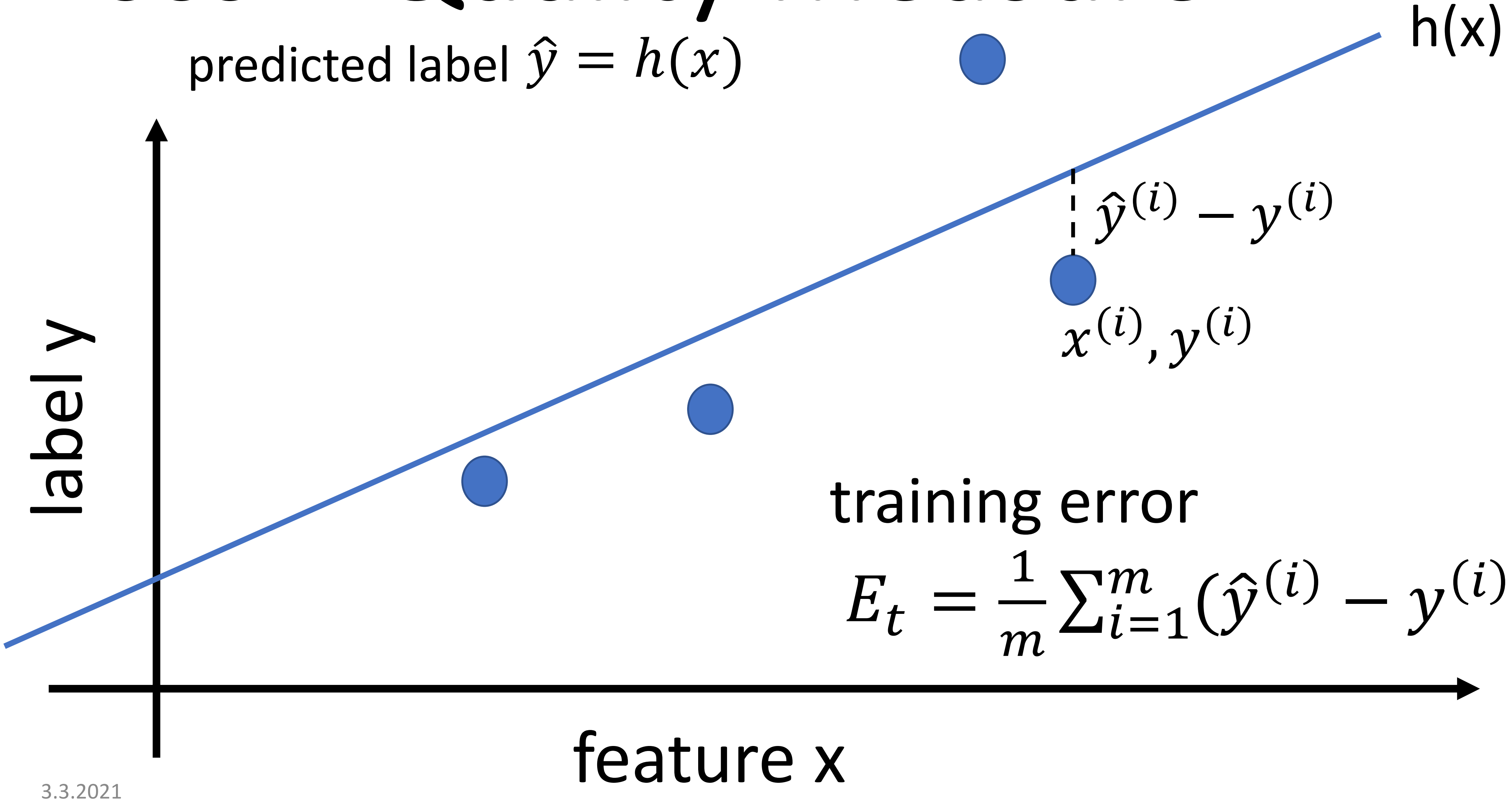
Model 4= “Nearest Neighbor”



3. Loss Function

Loss = Quality Measure

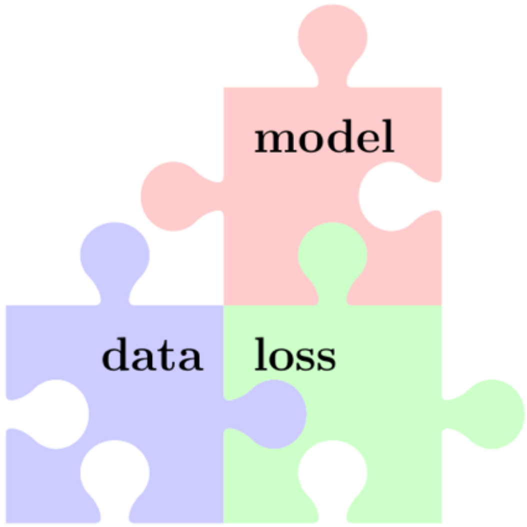
predicted label $\hat{y} = h(x)$



training error

$$E_t = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

Putting Together the Pieces



Plethora of ML methods obtained by combining difference choices for data representation, model and loss function

see Chapter 3 of mlbook.cs.aalto.fi

What is ML ?

informal: learn hypothesis out of a hypothesis space or “model” that incurs minimum loss when predicting labels of datapoints based on their features

formal:

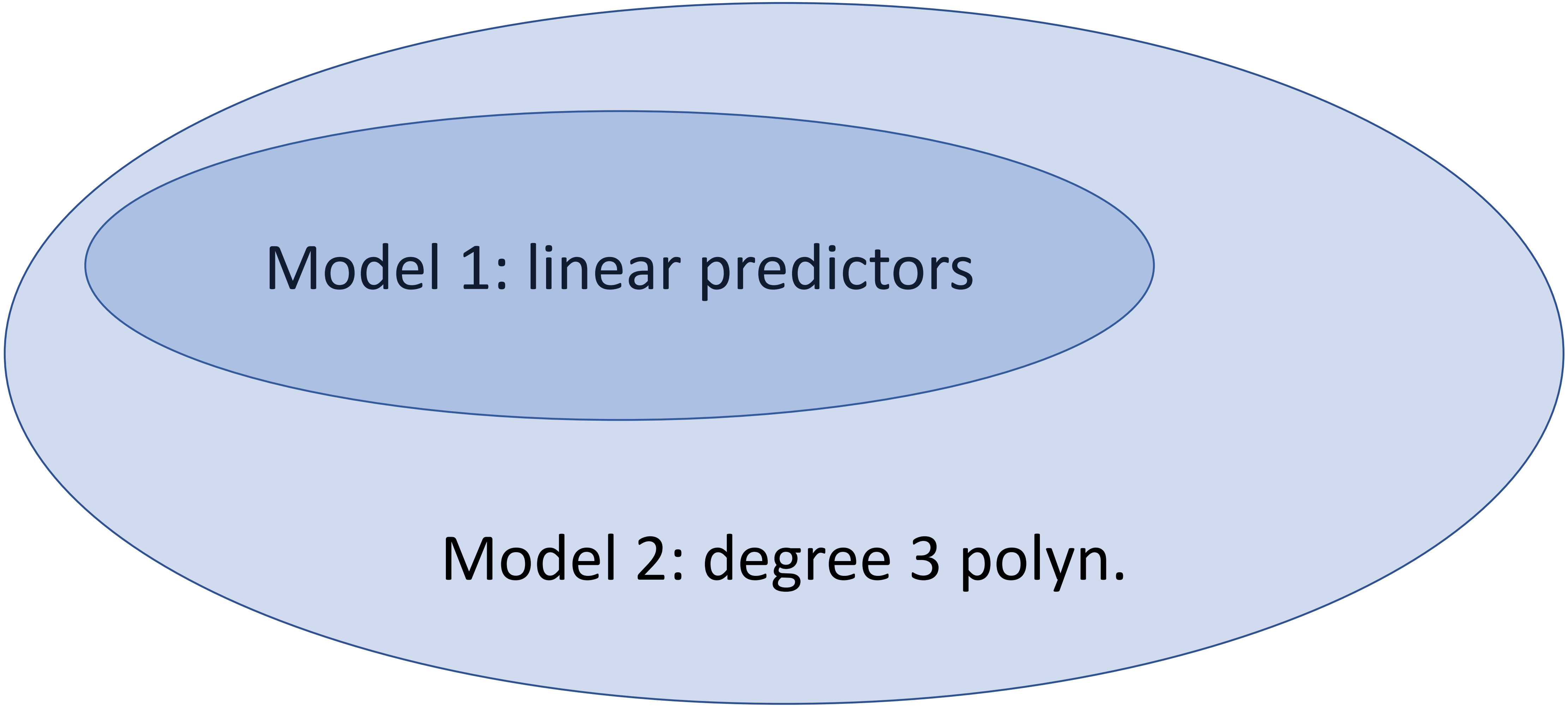
$$
\begin{aligned}
\hat{h} &= \operatorname*{argmin}_{h \in \mathcal{H}} \mathcal{E}(h|\mathcal{D}) \\
&\overset{(\boxed{2.12})}{=} \operatorname*{argmin}_{h \in \mathcal{H}} (1/m) \sum_{i=1}^{m} \mathcal{L}((\mathbf{x}^{(i)}, y^{(i)}), h).
\end{aligned}
$$

see Ch. 4.1 of mlbook.cs.aalto.fi

ML Methods and their Models

- linear regression uses linear hypothesis maps
- logistic regression uses linear hypothesis maps
- decision trees uses maps represented by flow-charts
- nearest neighbors uses piece-wise constant maps

Nested Models



Nested Models

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

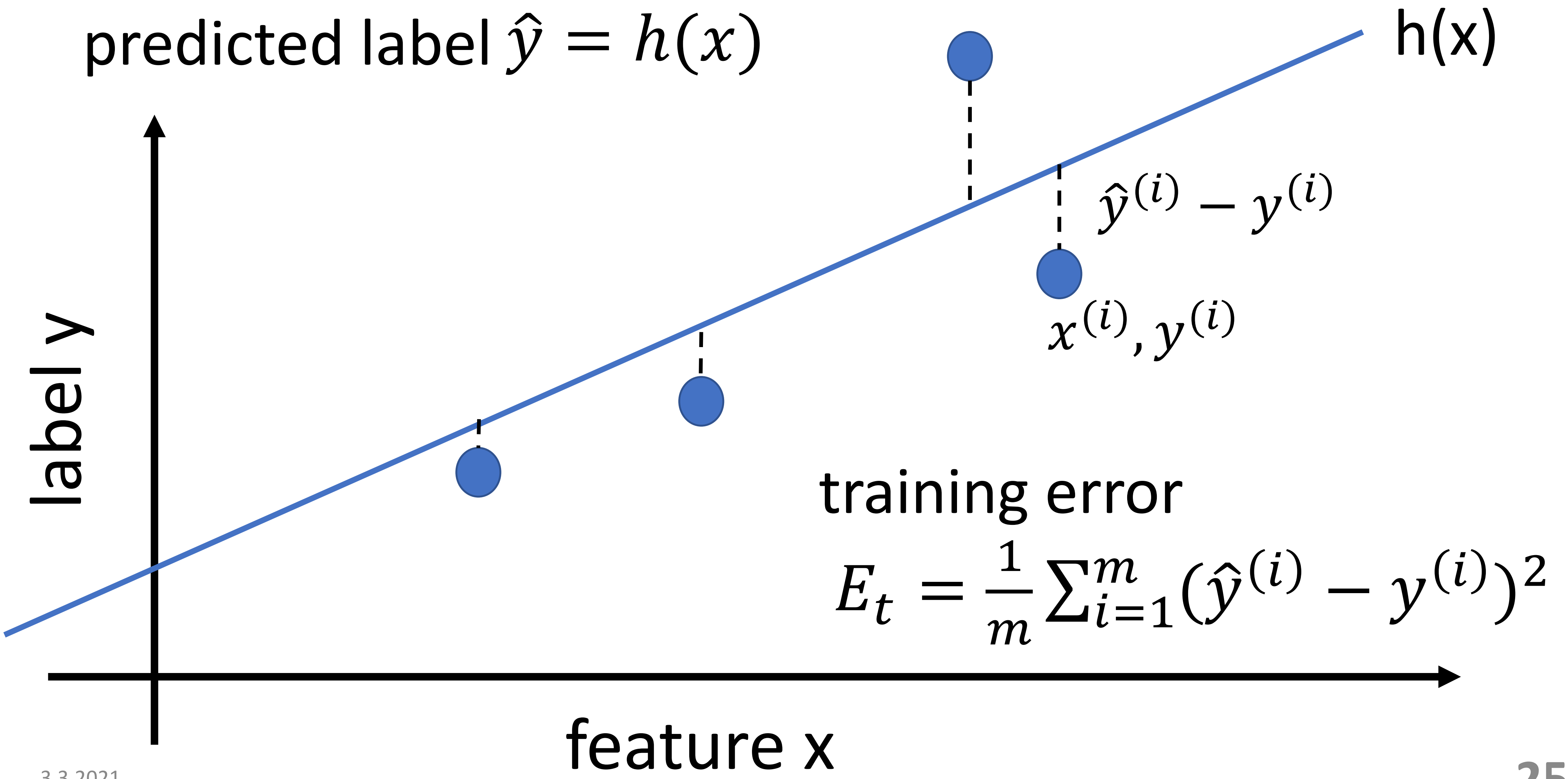
$\mathcal{H}^{(1)}$ … linear hypotheses

$\mathcal{H}^{(4)}$ … degree 4 polyn.

$\mathcal{H}^{(1)} \subset \mathcal{H}^{(2)} \subset \mathcal{H}^{(3)} \subset \mathcal{H}^{(4)} \subset \ldots$

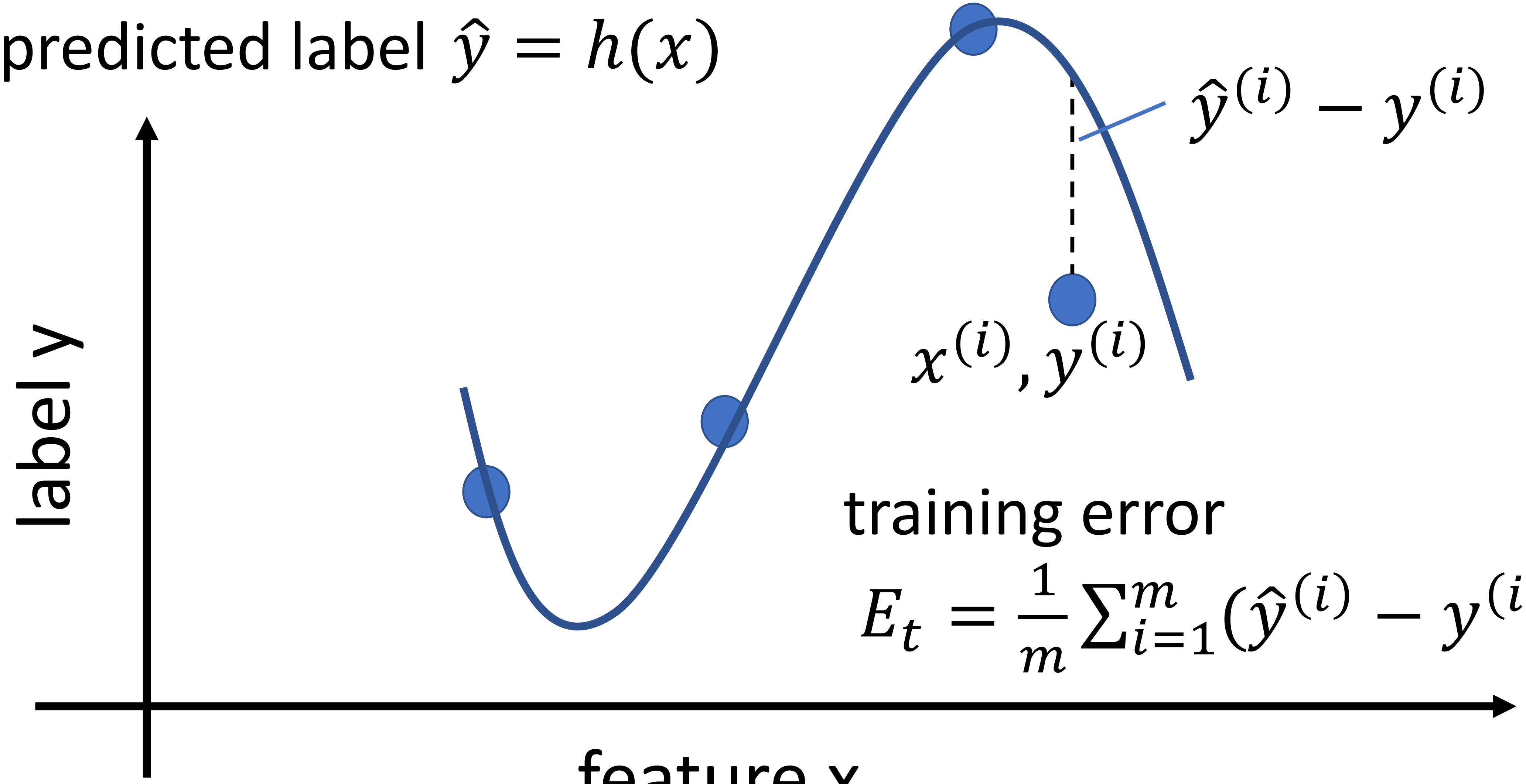
which model should we use ?

Model 1= “h(x) polynomial w. degree 1”



Model 2 = “h(x) polynomial with degree 3”

predicted label $\hat{y} = h(x)$



training error

$$E_t = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

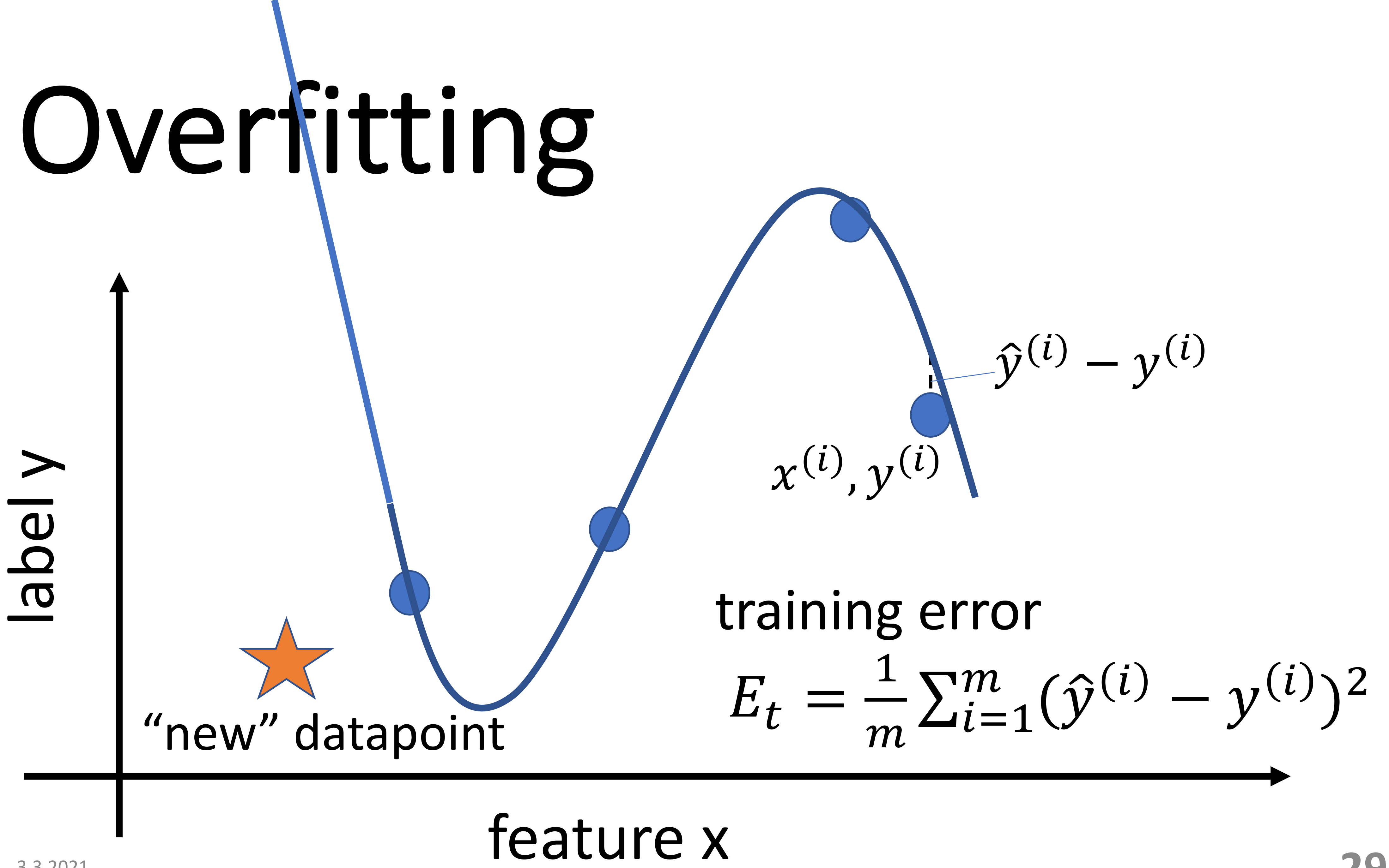
Training Errors

model 1
degree 1 polyn.

model 2:
degree 3 polyn.

small train error does not guarantee good performance outside the training set!

Overfitting



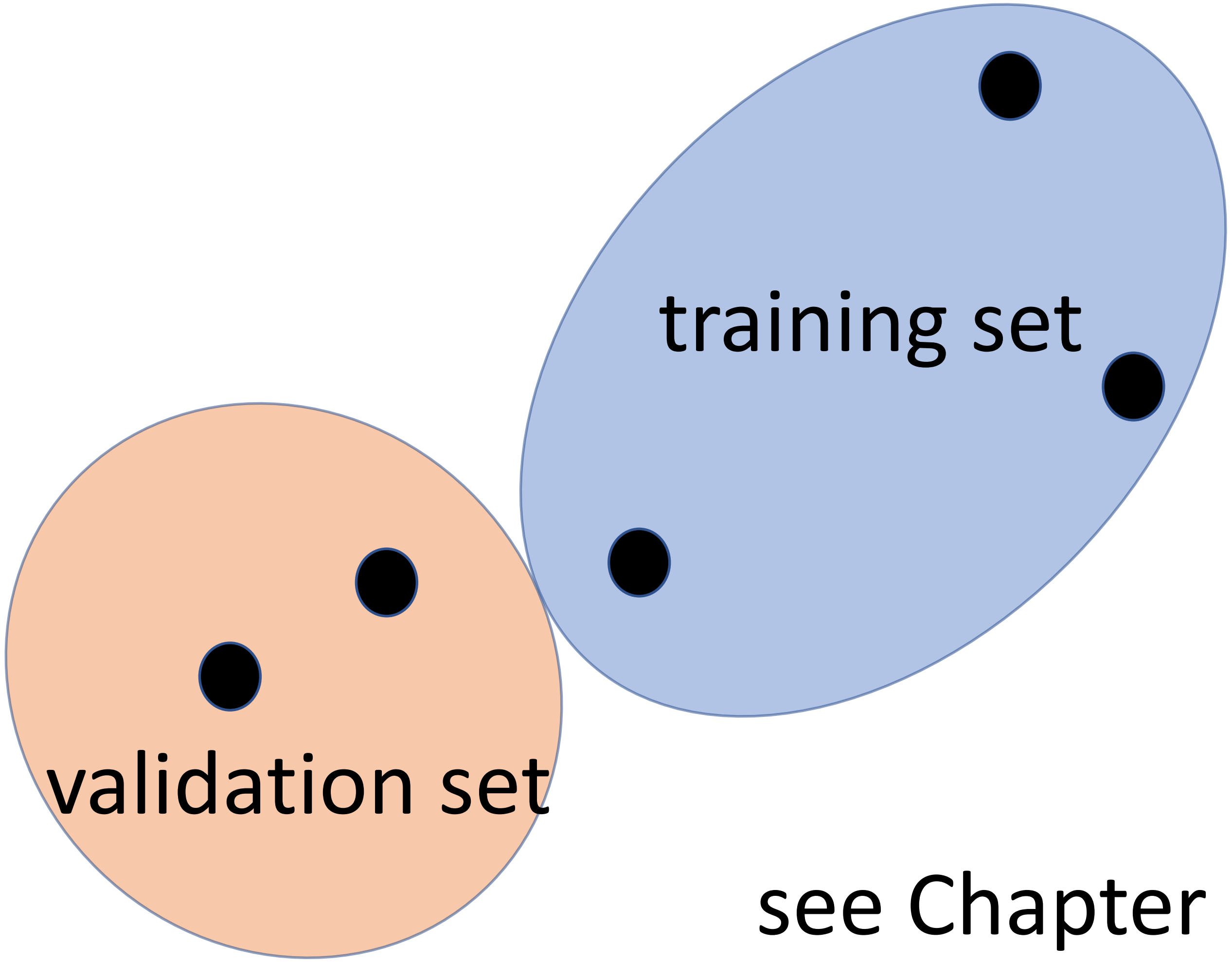
small training error only indicates that we have solved the ERM optimization problem

Model Validation and Selection

Learn and Validate!

- divide datapoints into two subsets
- training and validation set
- train.set: used to learn a hypothesis $\hat{h}$
- val.set: used to probe $\hat{h}$ outside trainset

Split Data into Training and Validation Set !



see Chapter 6.2 of mlbook.cs.aalto.fi

Python library:

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

Split into Train and Val Set in Python

labels of datapoints in trainset

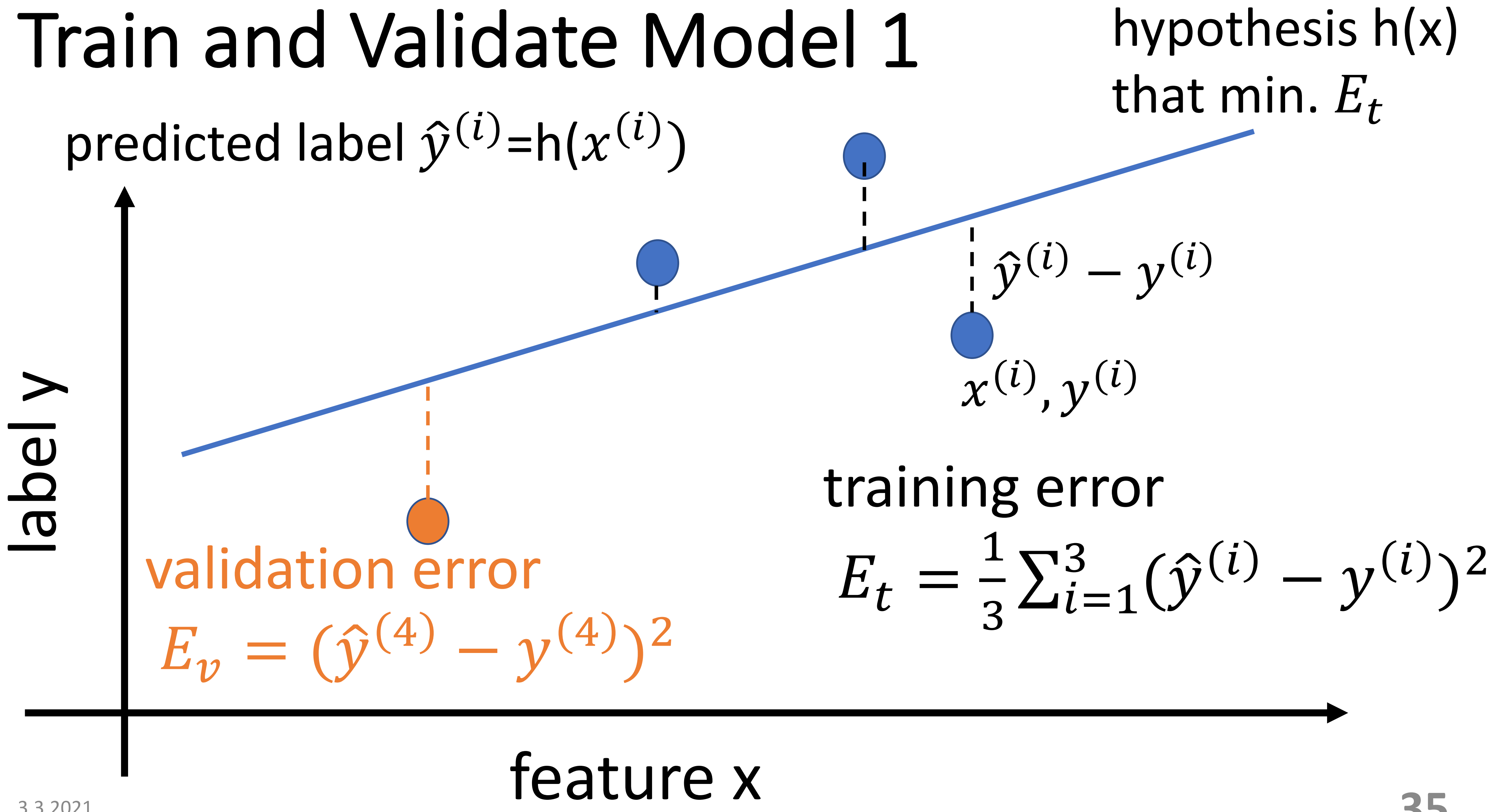
labels of datapoints

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

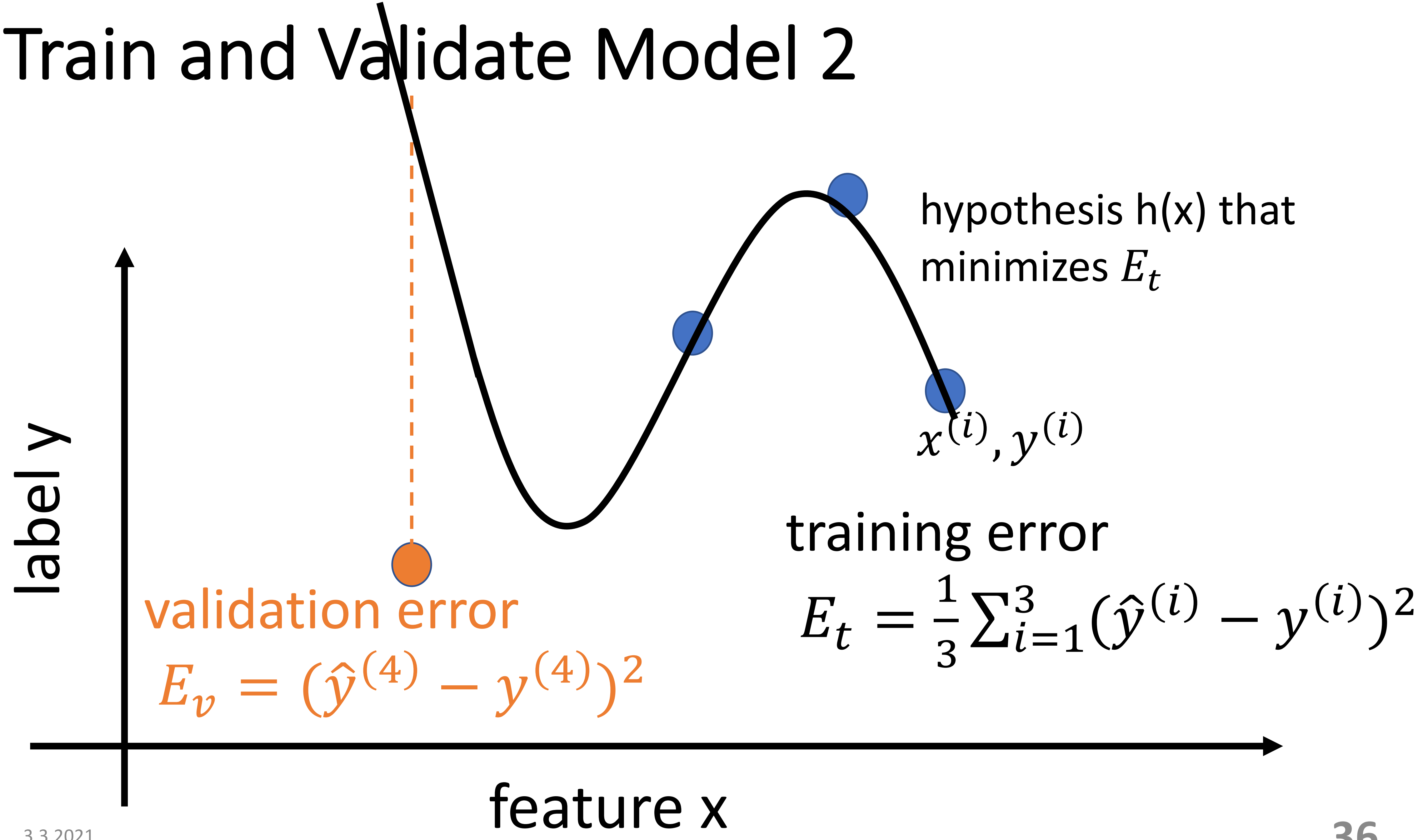
feature vectors of datapoints in trainset

feature vectors of datapoints

Train and Validate Model 1



Train and Validate Model 2



Train and Validate in Python

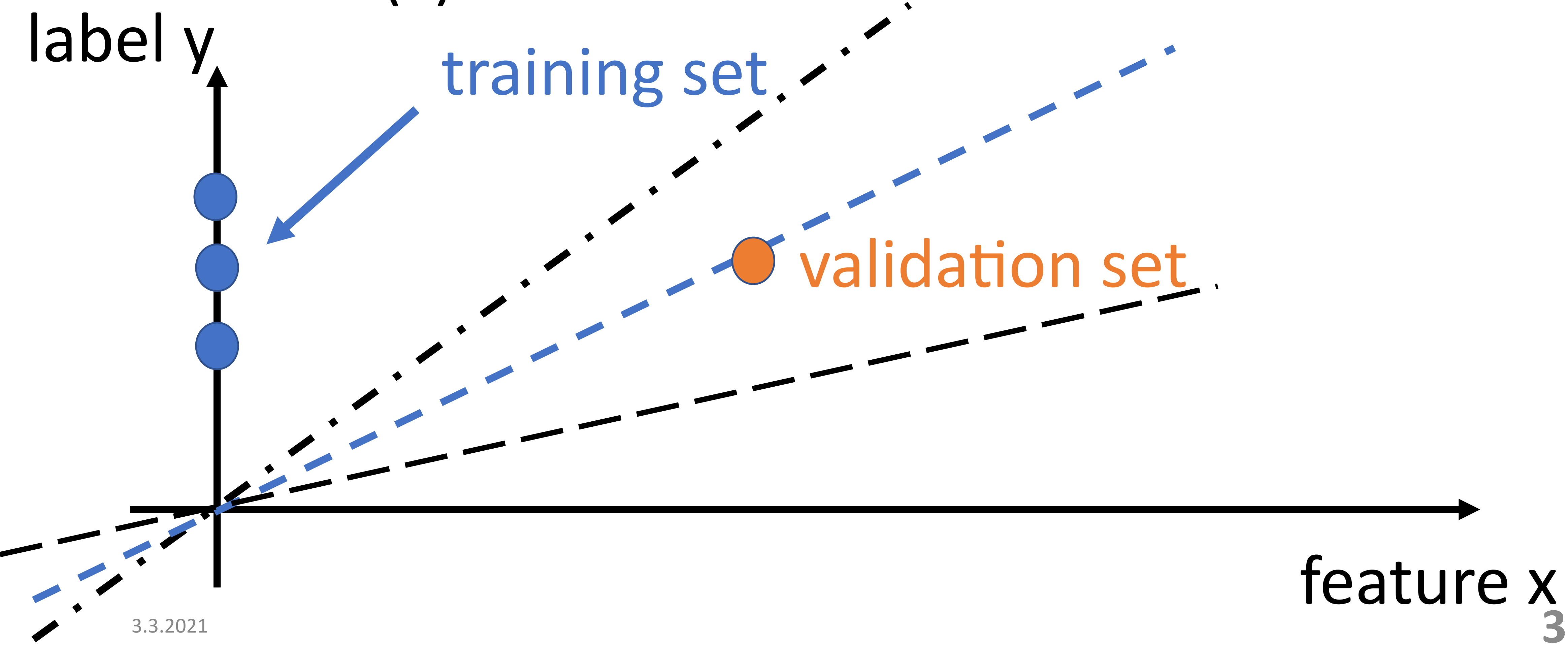
```
model.fit(X_train, y_train)
training_error = mean_squared_error(y_train,model.predict(X_train))
validation_error = mean_squared_error(y_val,model.predict(X_val))
```

K-Fold Cross Validation

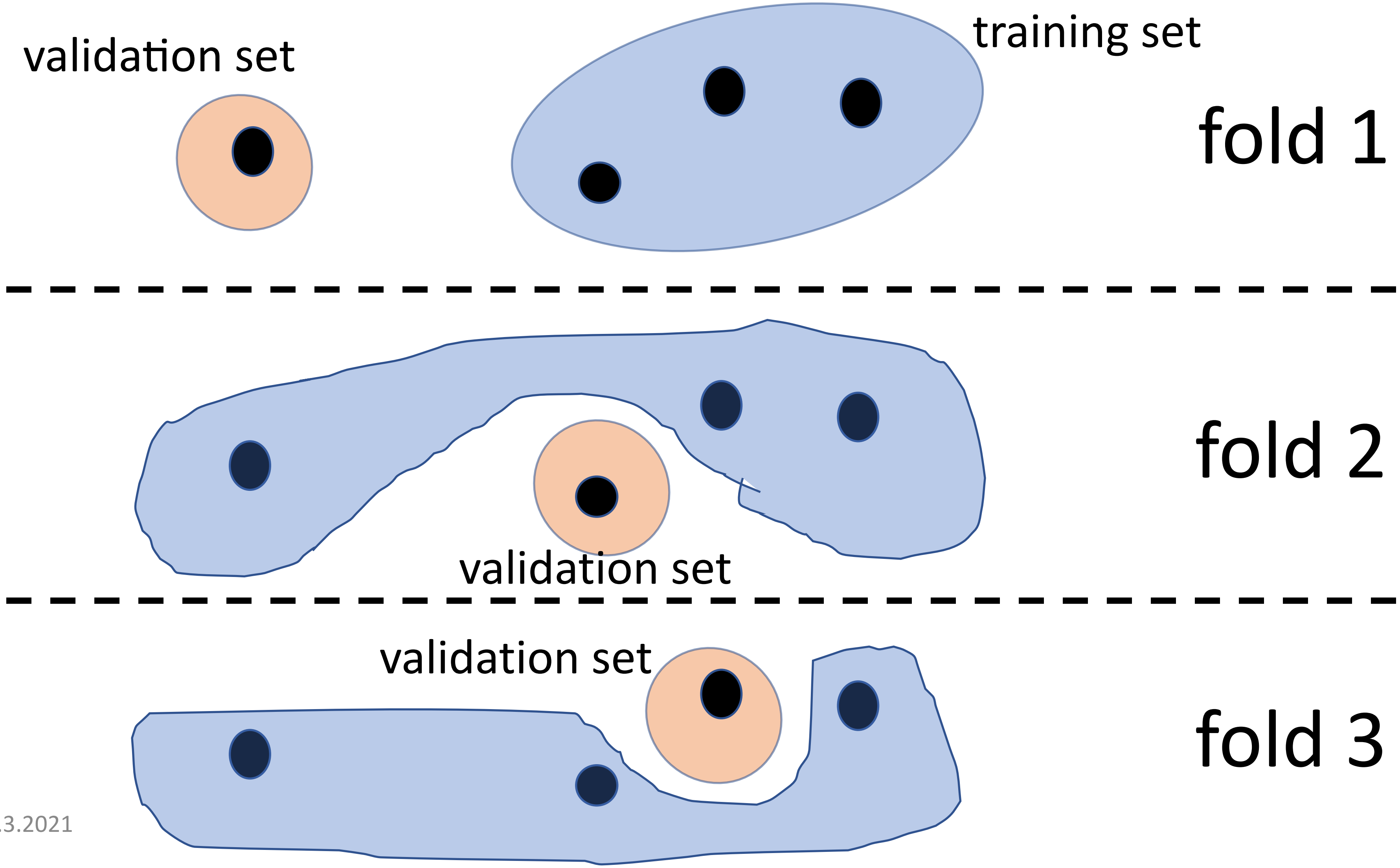
- might be unlucky with train/val split
- problematic for small datasets
- IDEA: randomly split several times
- “average out” unlucky splits

Unlucky Split into Train and Val Set

- model: h(x) = w*x

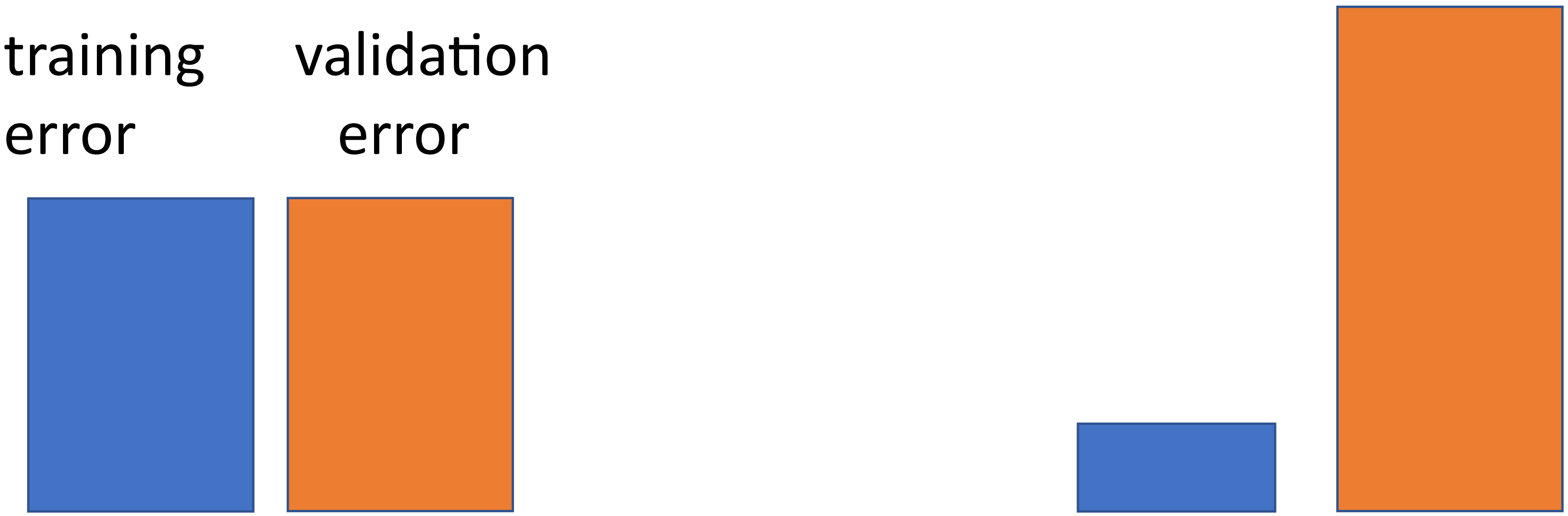


K-Fold Cross Validation



Basic Idea of Model Selection

- choose model with smallest validation error!



model 1
degree 1 polyn.

model 2:
degree 3 polyn.

Use Different Loss for Train and Val

- we can use different loss for training and validation
- this enables the comparison of different ML methods
- logistic regression uses log loss to learn hypothesis h1(x)
- SVM uses hinge loss to learn hypothesis h2(x)
- compare h1, h2 by their average 0/1 loss (“accuracy”) on val. set

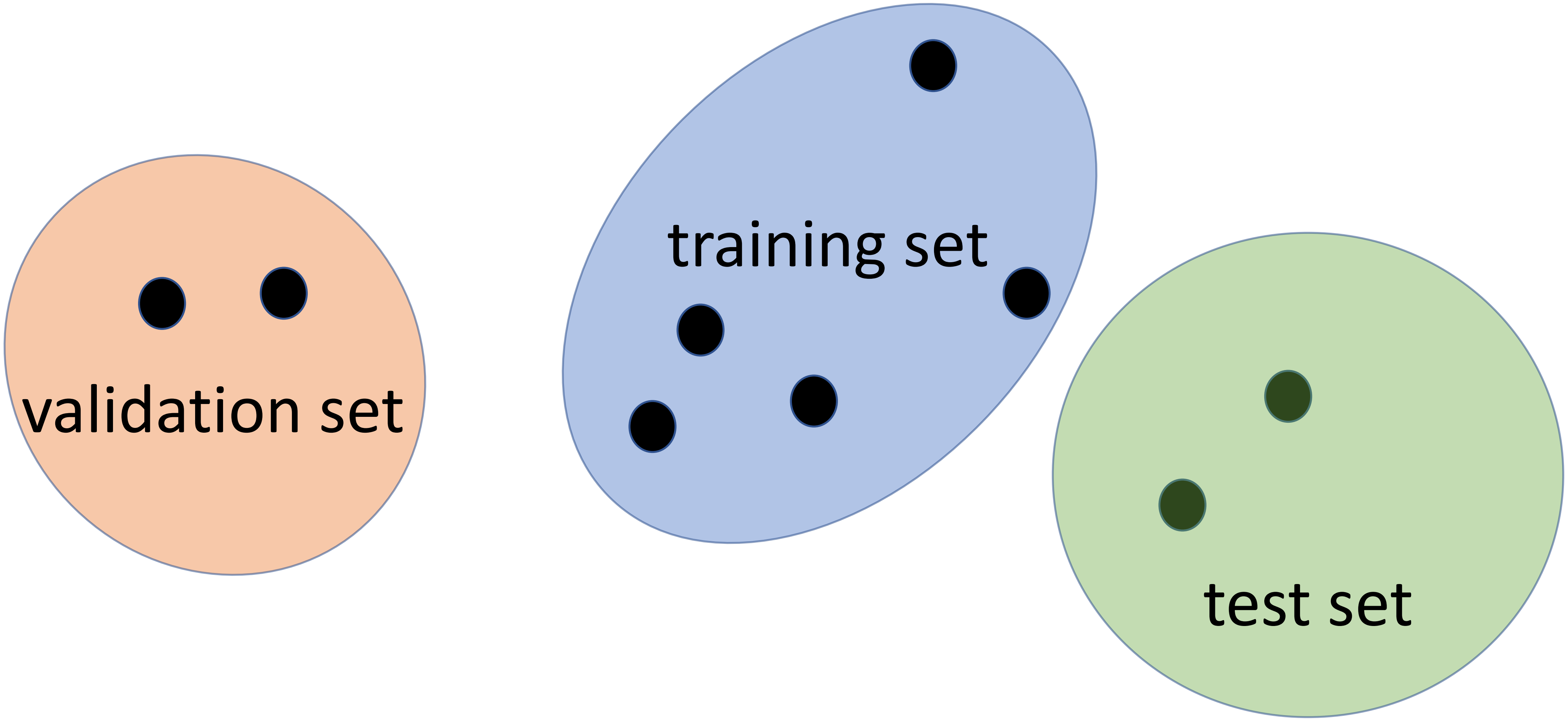
Compare Log.Reg. and SVM in Python

```python
# split dataset into training and validation set
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=42)

# learn hypothesis h1(x) by minimizing its average logistic loss on the training set
logreg= LogisticRegression(random_state=0).fit(X_train, y_train)
# compute validation error of learnt hypothesis h1(x) using average 0/1 loss (accuracy)
val_error_logreg = logreg.score(X_val, y_val)

svmclf = svm.SVC()
#learn hypothesis h2(x) by minimizing its average hinge loss on the training set
svmclf.fit(X_train, y_train)
# compute validation error of learnt hypothesis h2(x) using average 0/1 loss (accuracy)
val_error_svm = svmclf.score(X_val, y_val)

print("accuracy of logistic regression on validation set:",val_error_logreg)
print("accuracy of logistic regression on validation set:",val_error_svm)
```

Test Set

- chosen model $\mathcal{H}^{(r)}$ with validation error E_v
- validation error E_v is a poor performance indicator
- E_v too optimistic since $\mathcal{H}^{(r)}$ choosen with smallest E_v
- need a test set different from train. and valset

Split Data into Train, Val and Test Set



Model Selection Recipe

- input: list of candidate models $\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \dots, \mathcal{H}^{(M)}$
- for each candidate model $\mathcal{H}^{(l)}, l = 1, \dots M$:
  - learn optimal hypothesis $\hat{h}^{(l)}$ by min. train. error
  - compute validation error $E_v^{(l)}$ on val. set
- choose $\hat{h}^{(r)} \in \mathcal{H}^{(r)}$ with min. val. error $E_v^{(r)} = min\, E_v^{(l)}$
- compute test error of $\hat{h}^{(r)}$ on test set

How Large should we choose the Validation Set ?

validation error is sum of individual loss terms

$$E_v = \frac{1}{m_v} \sum_{i=1}^{m_v} \underbrace{\left(h(x^{(i)}) - y^{(i)}\right)^2}_{}$$

loss $L^{(i)}$ for i-th datapoint

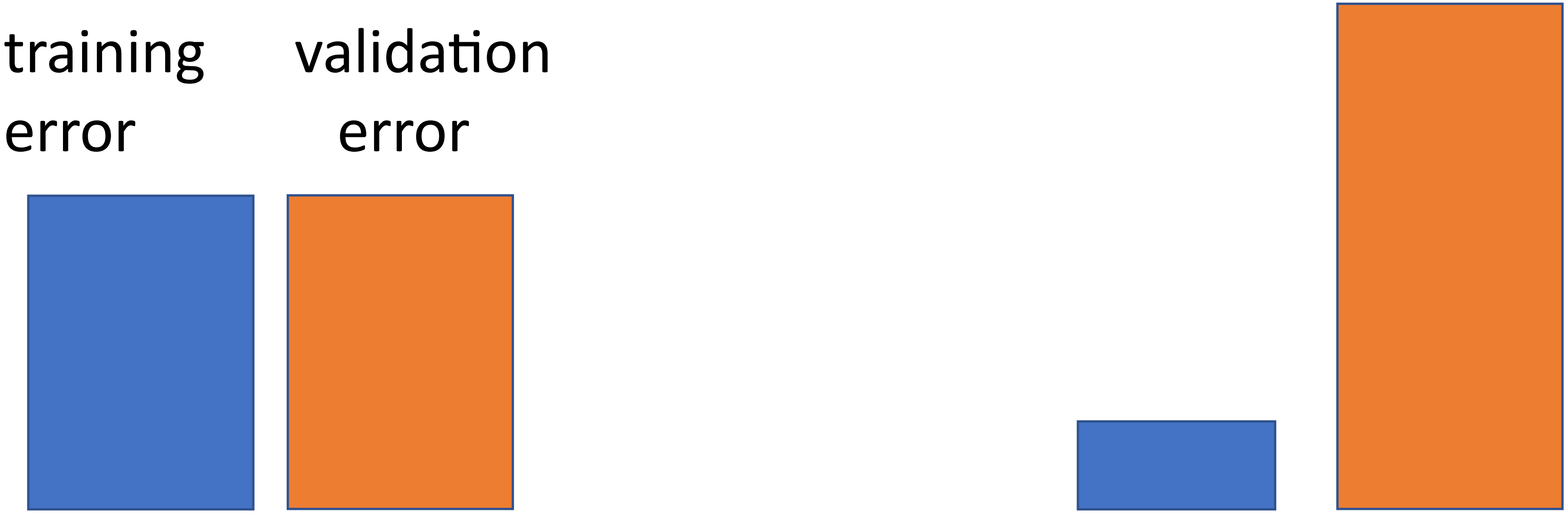
interpret data as realizations of i.i.d. random variables (RV),
=> loss values $L^{(i)}$ become realizations of RVs

validation error E_v also becomes realization of RV with
mean $\mathrm{E}\{E_v\}$ = $\mathrm{E}\{L^{(i)}\}$ and variance $\sigma_v^2 = \frac{1}{m_v} \sigma_i^2$
with σ_i^2 being the (common) variance of $L^{(i)}$

The Effective Dimension of a Model

Basic Idea of Model Selection

- choose model with smallest validation error!



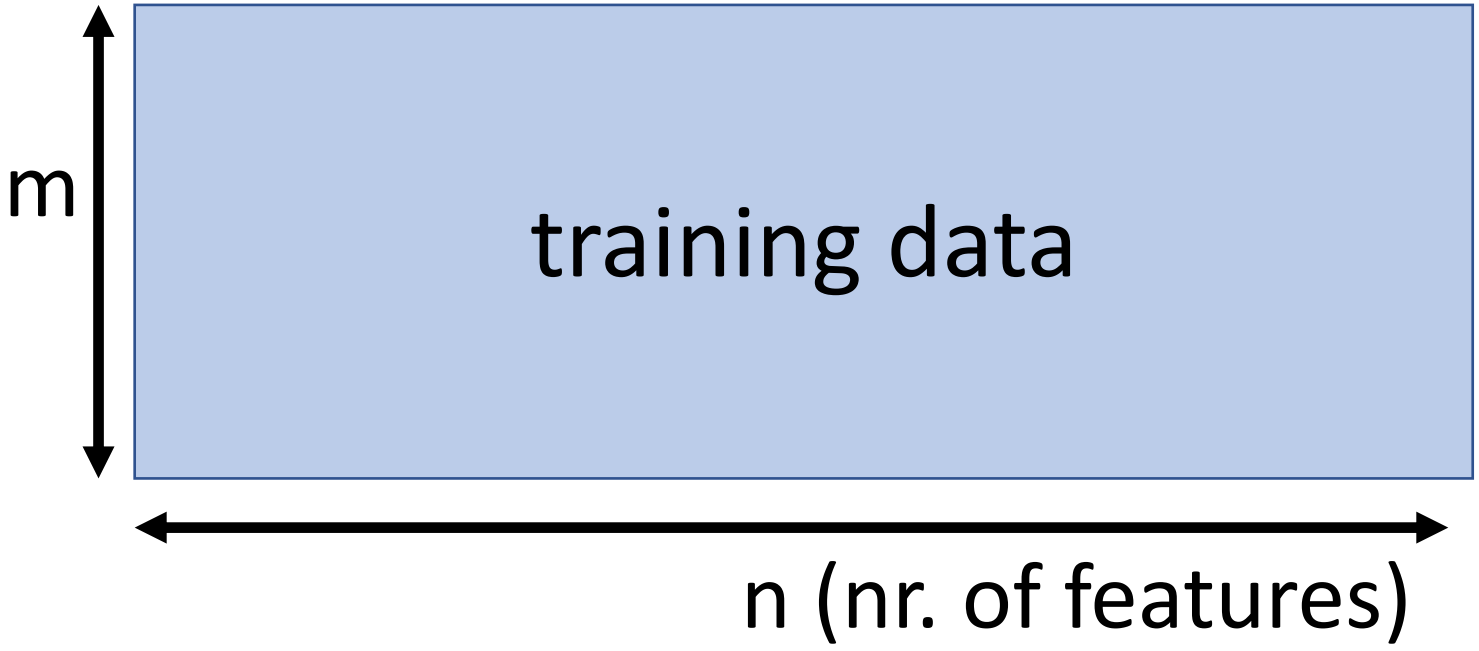
model 1
degree 1 polyn.

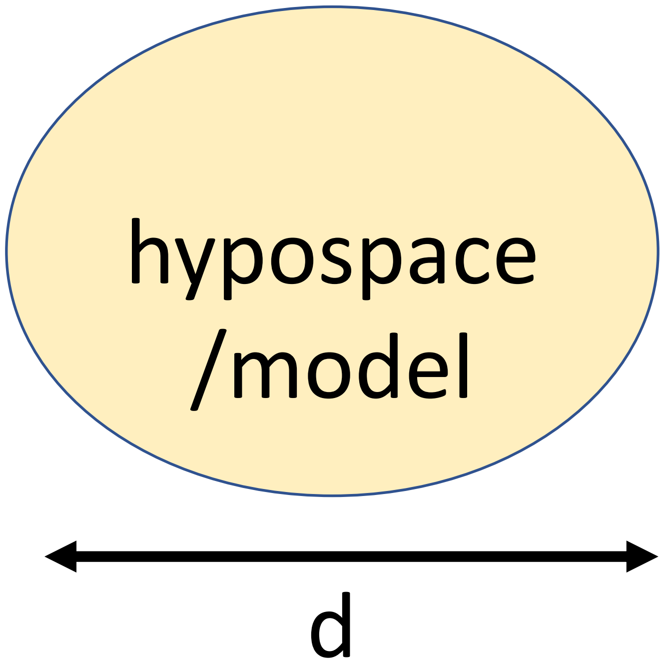
model 2:
degree 3 polyn.

Effective Dimension

- we define effective dimension d of a hypothesis space as maximum number of datapoints that can be perfectly fit by some hypothesis in that space
- provides a measure for size of hypospace!

Data and Model Size





crucial parameter is the ratio d/m

Effective Dim. Linear Maps

- linear map can perfectly fit m data points with n features, as soon as $n \geq m$ (see Ch 6.1)
- eff.dim. of linear maps = nr. of features
- $d = n$

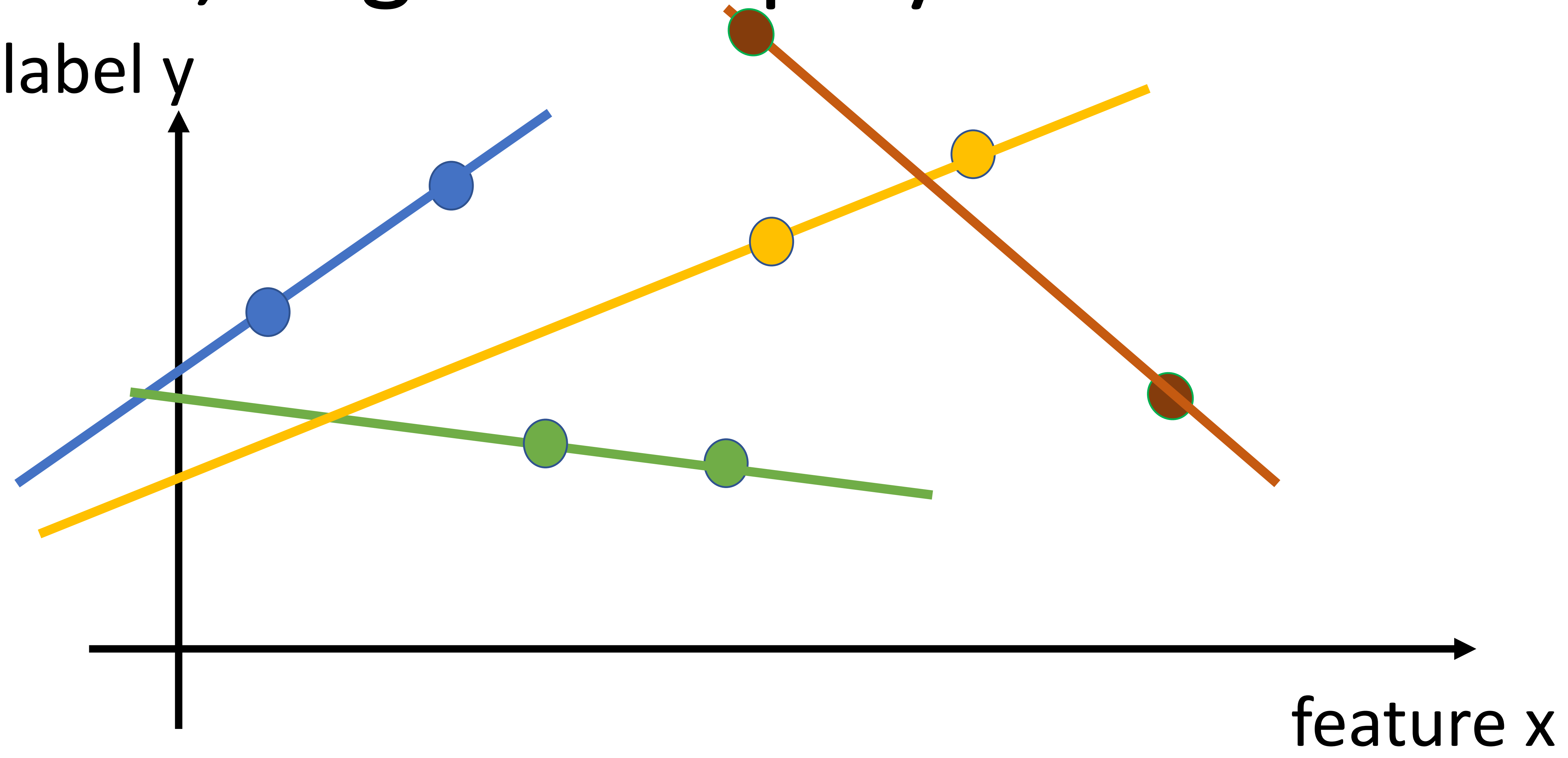
Effective Dim. Polyn. Reg.

perfectly fit (almost) any m data points using polynomials of max degree r as soon as

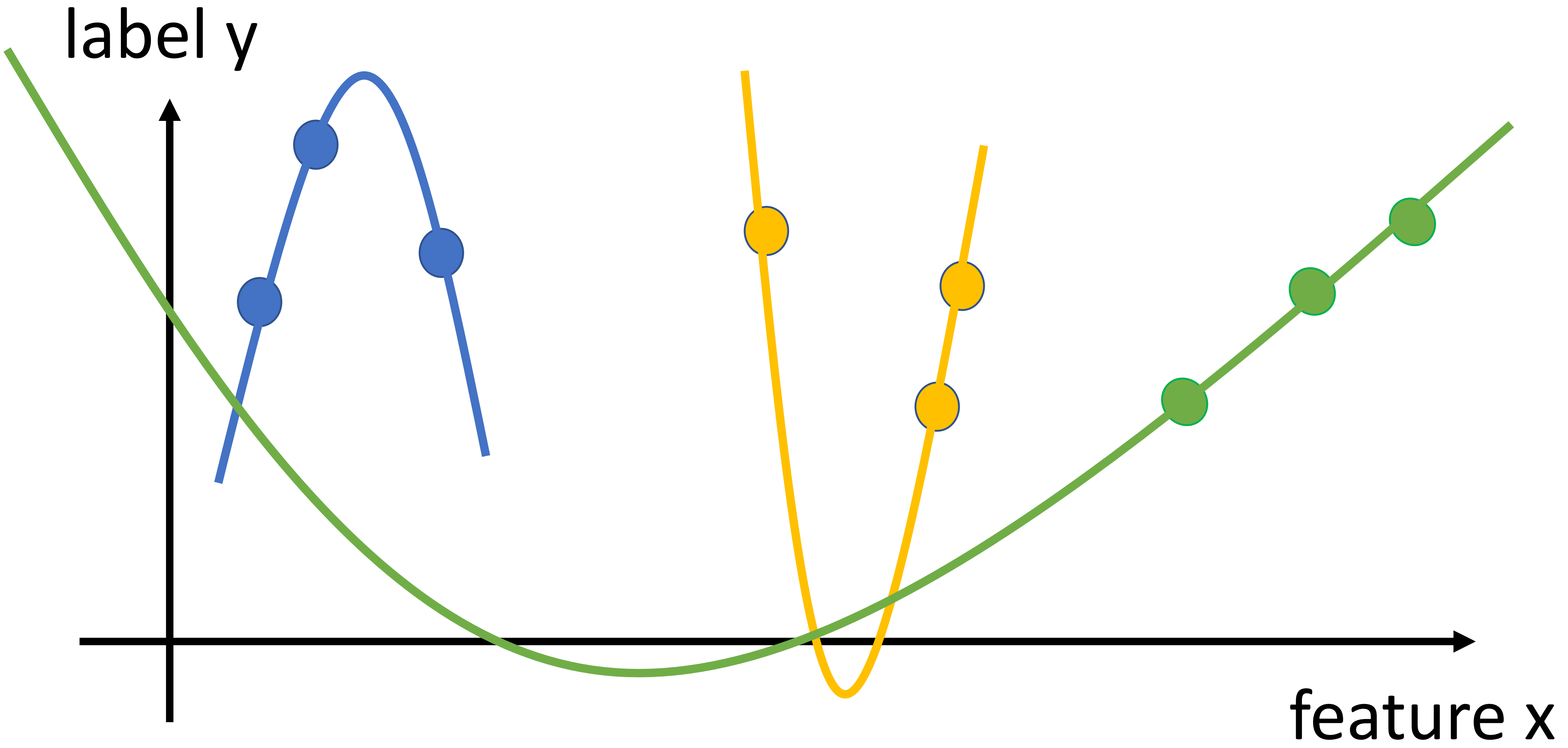
$$r+1 \geq m$$

-> $d = r+1$ (effective dim. of polyn. regression equals the max. polyn. degree plus one!)

m=2, degree r=1 polynomial

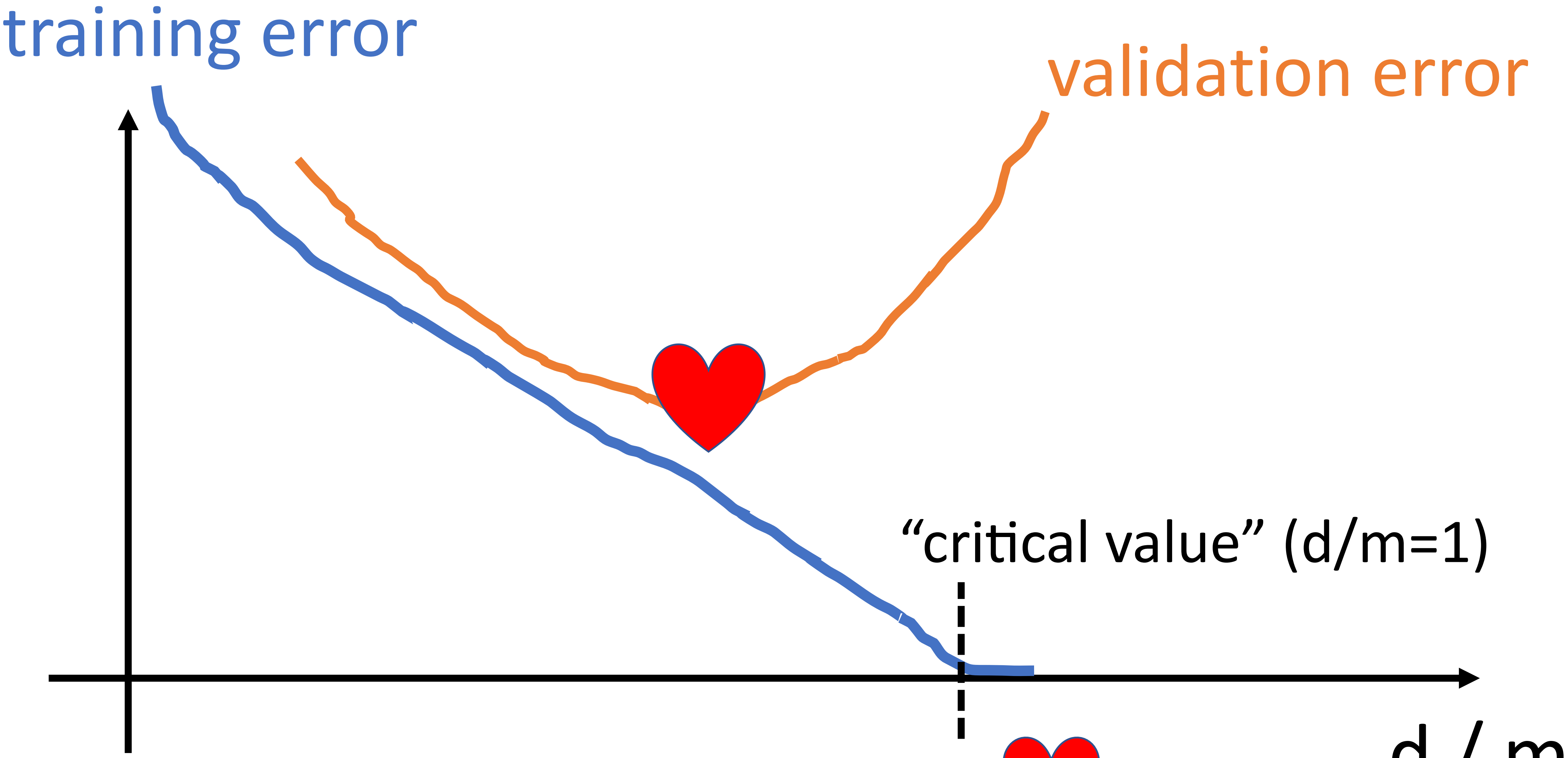


m=3, degree r=2 polynomial



Data Hungry ML Methods

- millions of features for datapoints (e.g. megapixel image)
- eff.dim. d of linear maps is also millions
- eff.dim d of deep nets is millions … billions
- can perfectly fit any set of 100000s (!) of datapoints
- training error will be zero (overfitting!)



adjust model and/or data to reach ♥

Take Home Messages

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

What’s Next ?

- so far, assume a given list of candidate models
- how to produce a good list of candidate models?
- lecture “Diagnosing ML” to find good candidate models
- lecture “Regularization” on manipulating “d/m”

Thank You !