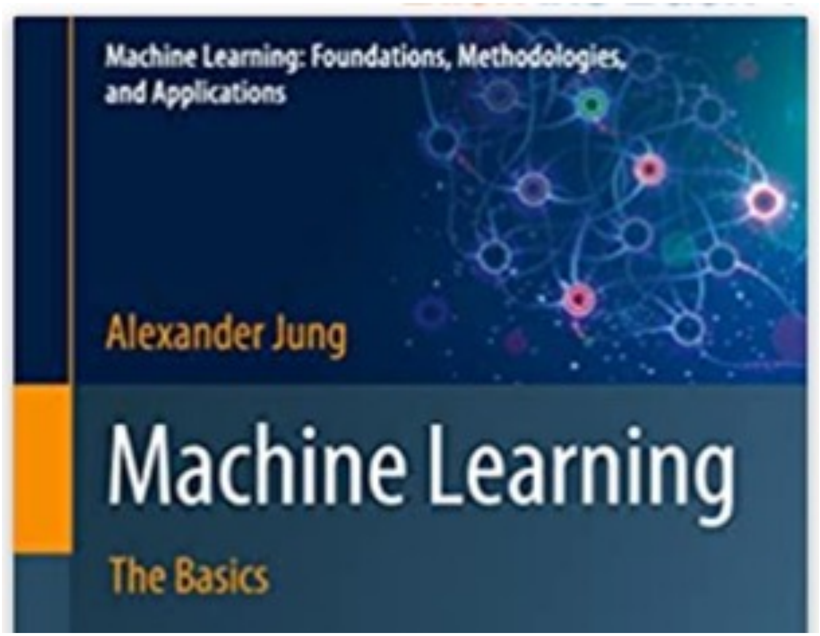
Regularization

Alex(ander) Jung
Assistant Professor for Machine Learning
Department of Computer Science
Aalto University

Reading.

Ch. 7 of https://mlbook.cs.aalto.fi



Learning Goals

- develop intuition for effective data and model size
- reduce model size by model pruning
- increase data size by data augmentation
- regularization = impl. model pruning = impl. data aug.
- use reg. for transfer - , multi-task – and semi-supervised learning

Empirical Risk Minimization

learn hypothesis out of model that incurs minimum loss when predicting labels of datapoints based on their features

training set

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

$$\overset{(\boxed{2.16})}{=} \operatorname*{argmin}_{h \in \mathcal{H}} (1/m) \sum_{i=1}^{m} L\big((\mathbf{x}^{(i)}, y^{(i)}), h\big).$$

hypothesis

label of i-th datapoint

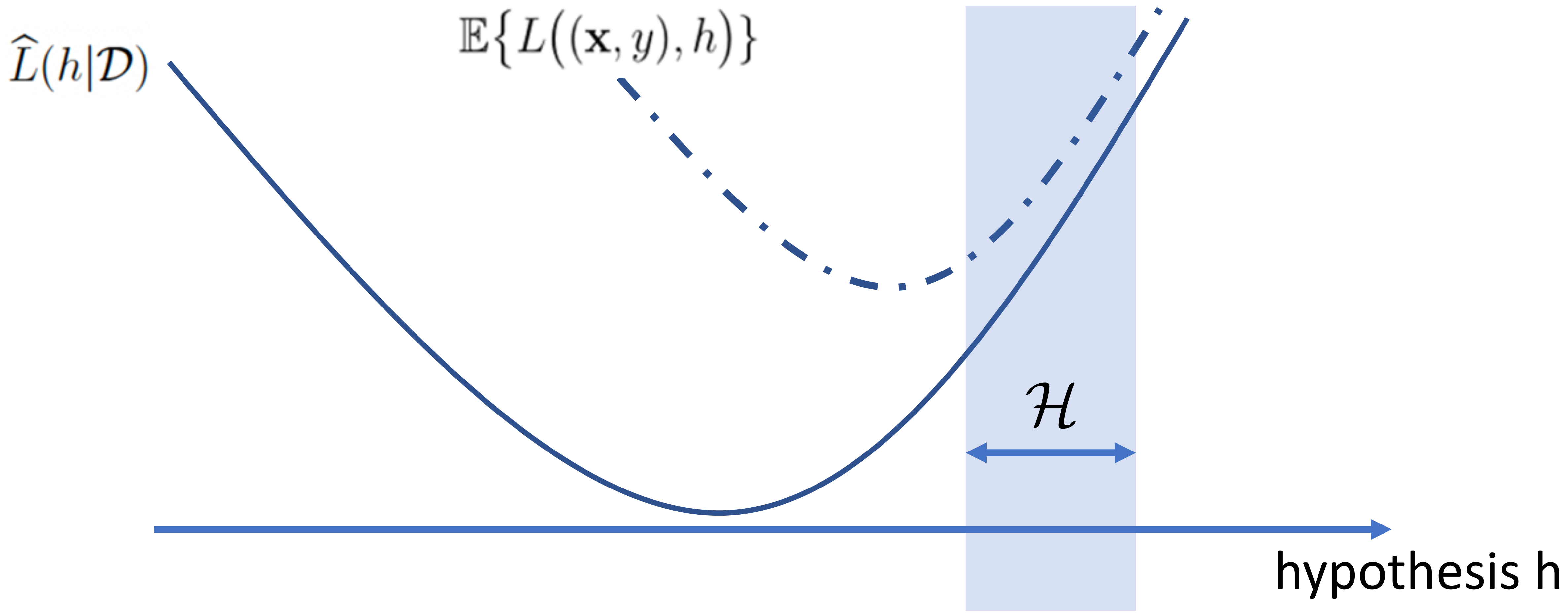
model

loss function

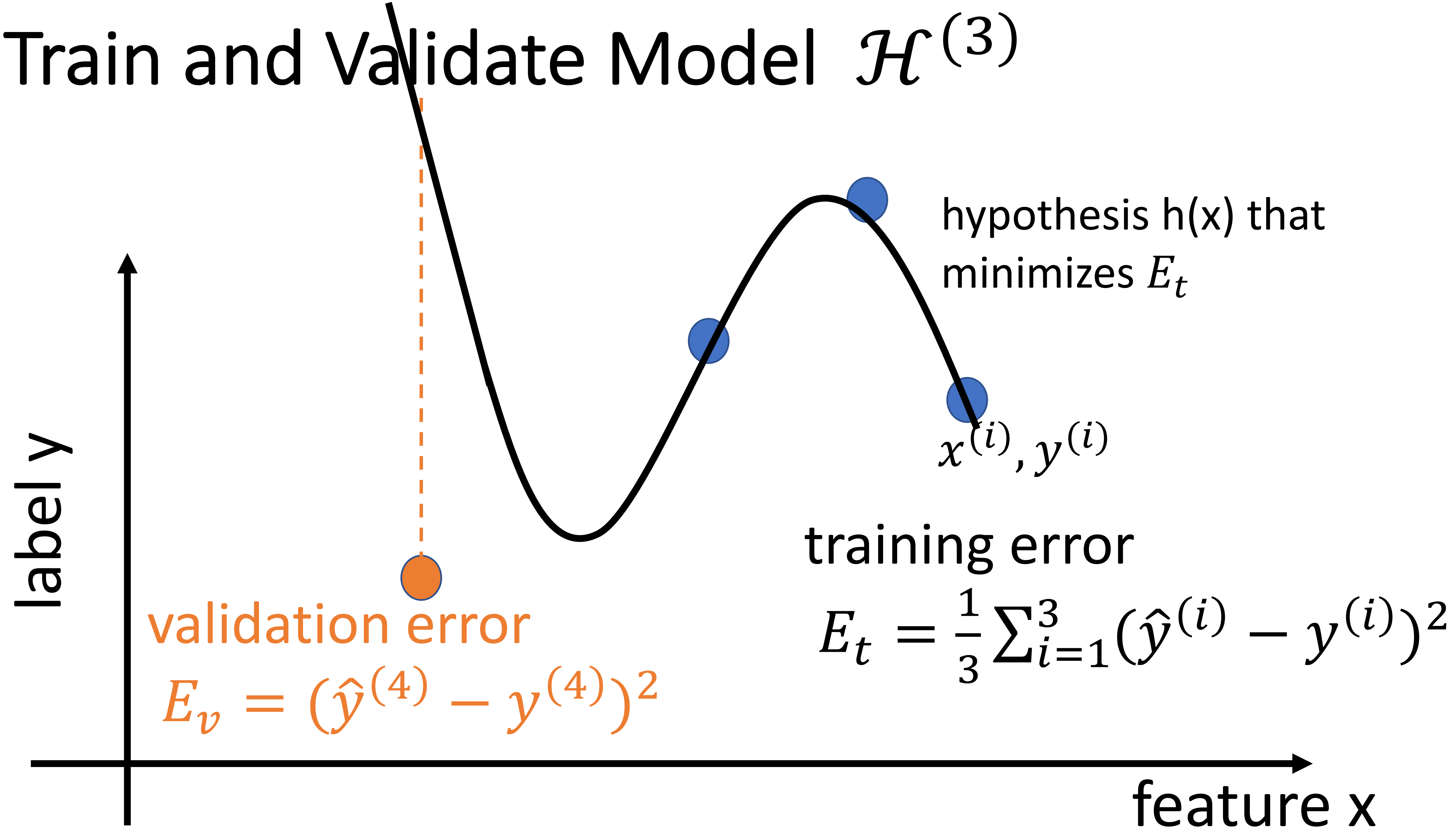
features of i-th datapoint

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

ERM is only Approximation!



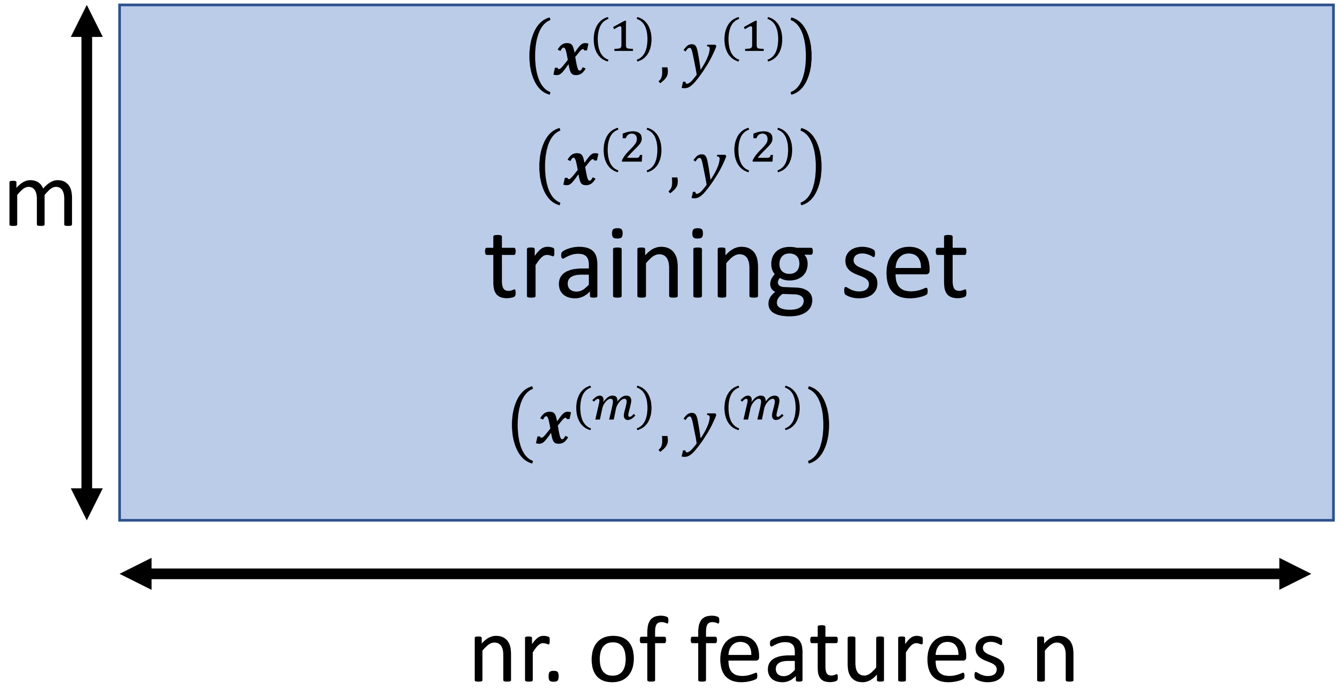
Train and Validate Model $\mathcal{H}^{(3)}$

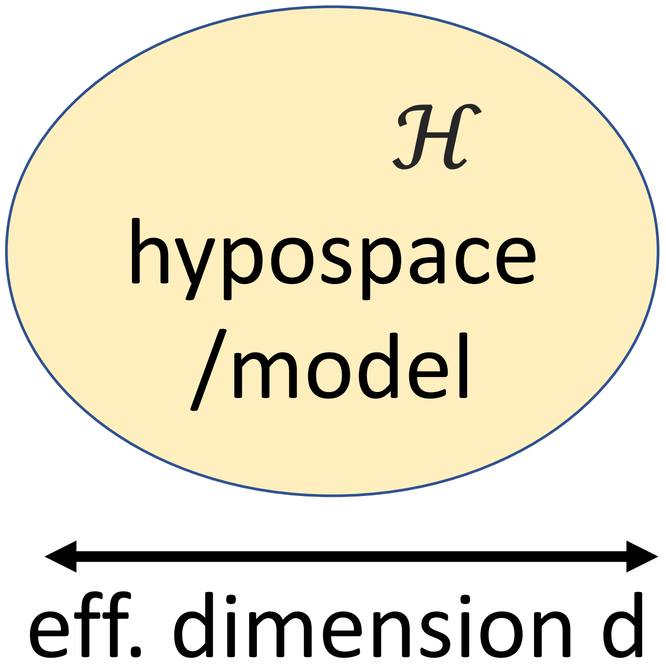


Small Training Error Does Not Imply Good Performance on New Data Points!

Small Training Error Merely Indicates That Optimization/Training Algorithm Works

Data and Model Size

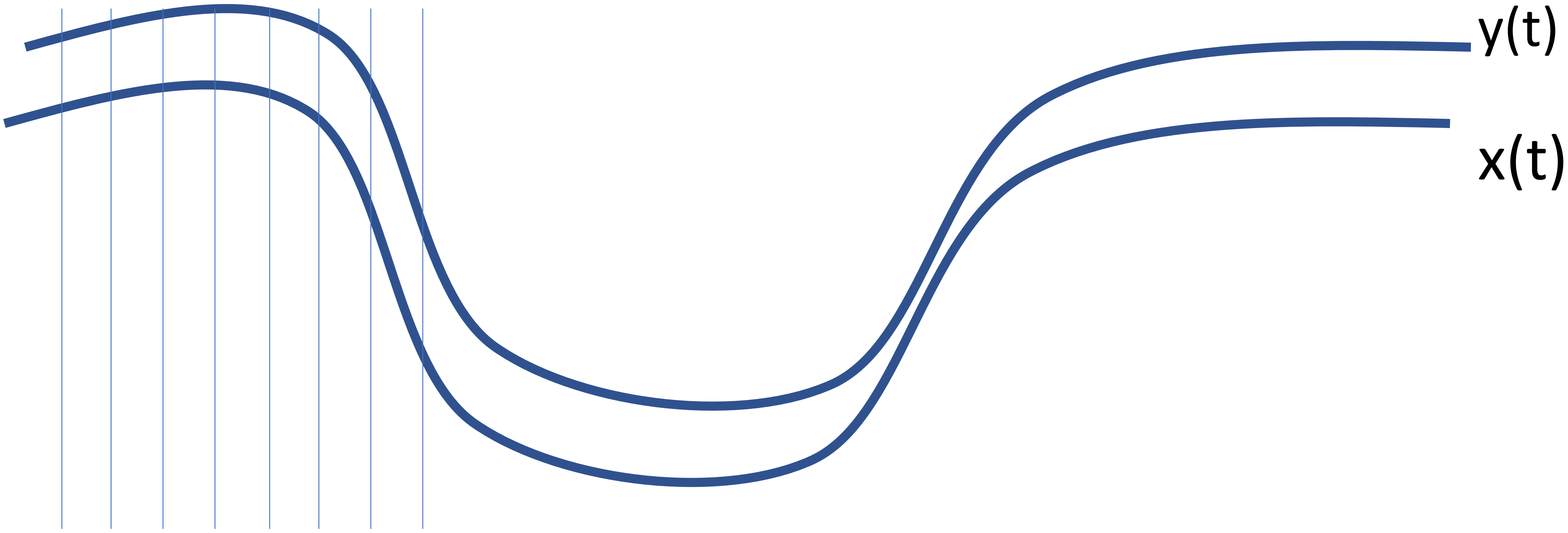




crucial parameter is the ratio d/m

Effective Data Size

consider data points obtained from time series



Effective Dim. Linear Maps

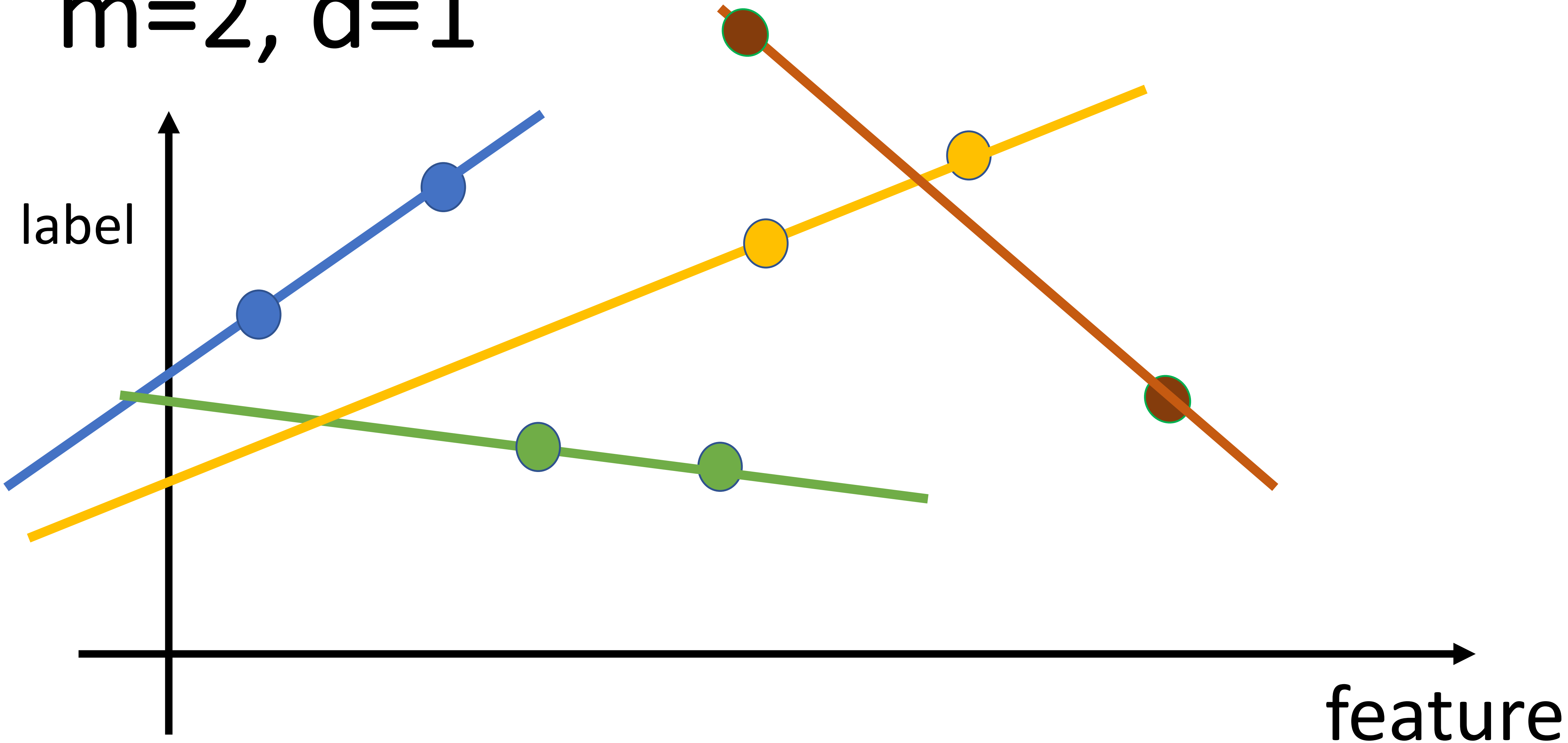
- linear map can perfectly fit m data points with n features, as soon as $n \geq m$ [Ch 6.1, mlbook.cs.aalto.fi]
- eff.dim. of linear maps = nr. of features
- $d = n$

Effective Dim. Linear Maps

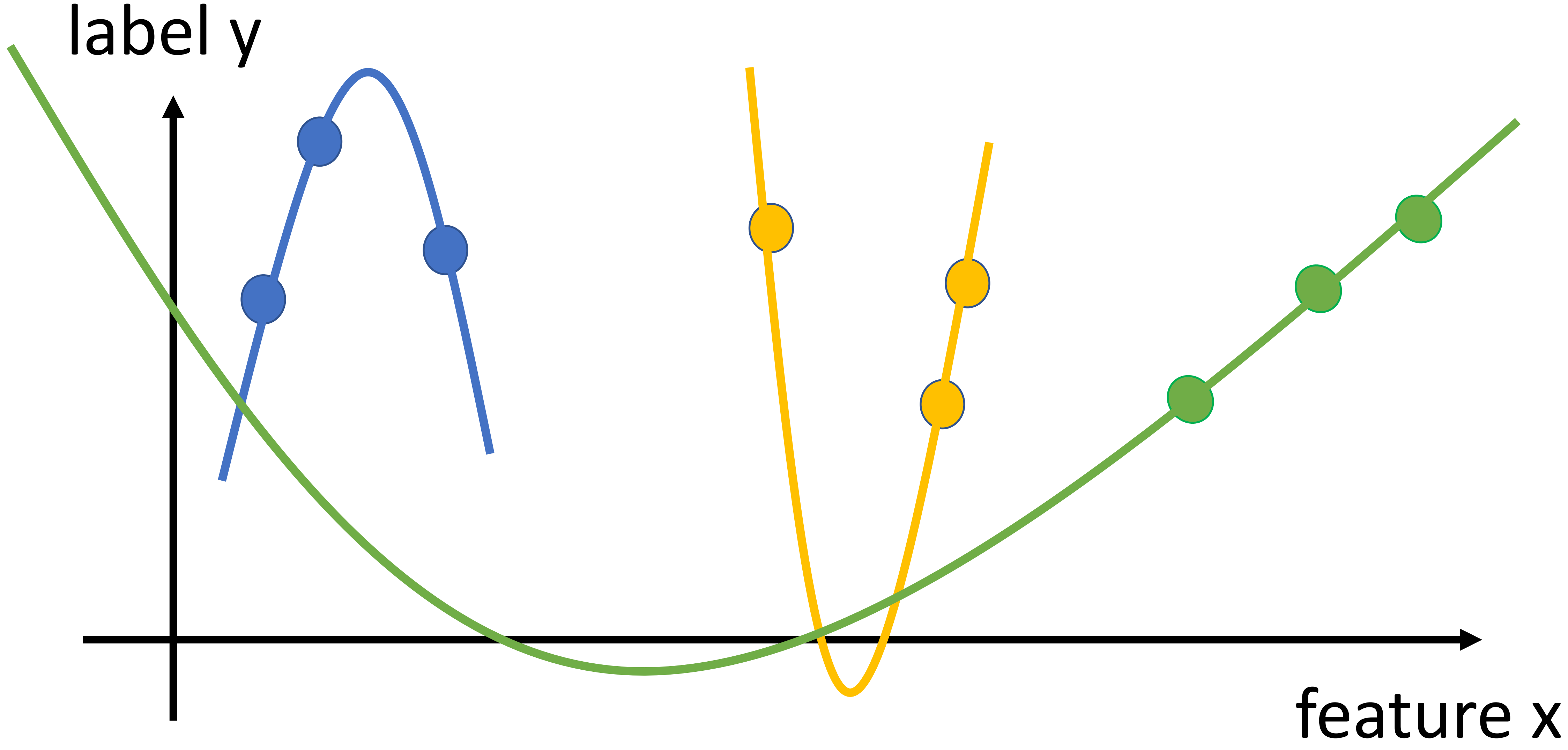
we can perfectly fit (almost) any m data points using polynomials of degree d as soon as

$$d \geq m\text{-}1$$

m=2, d=1

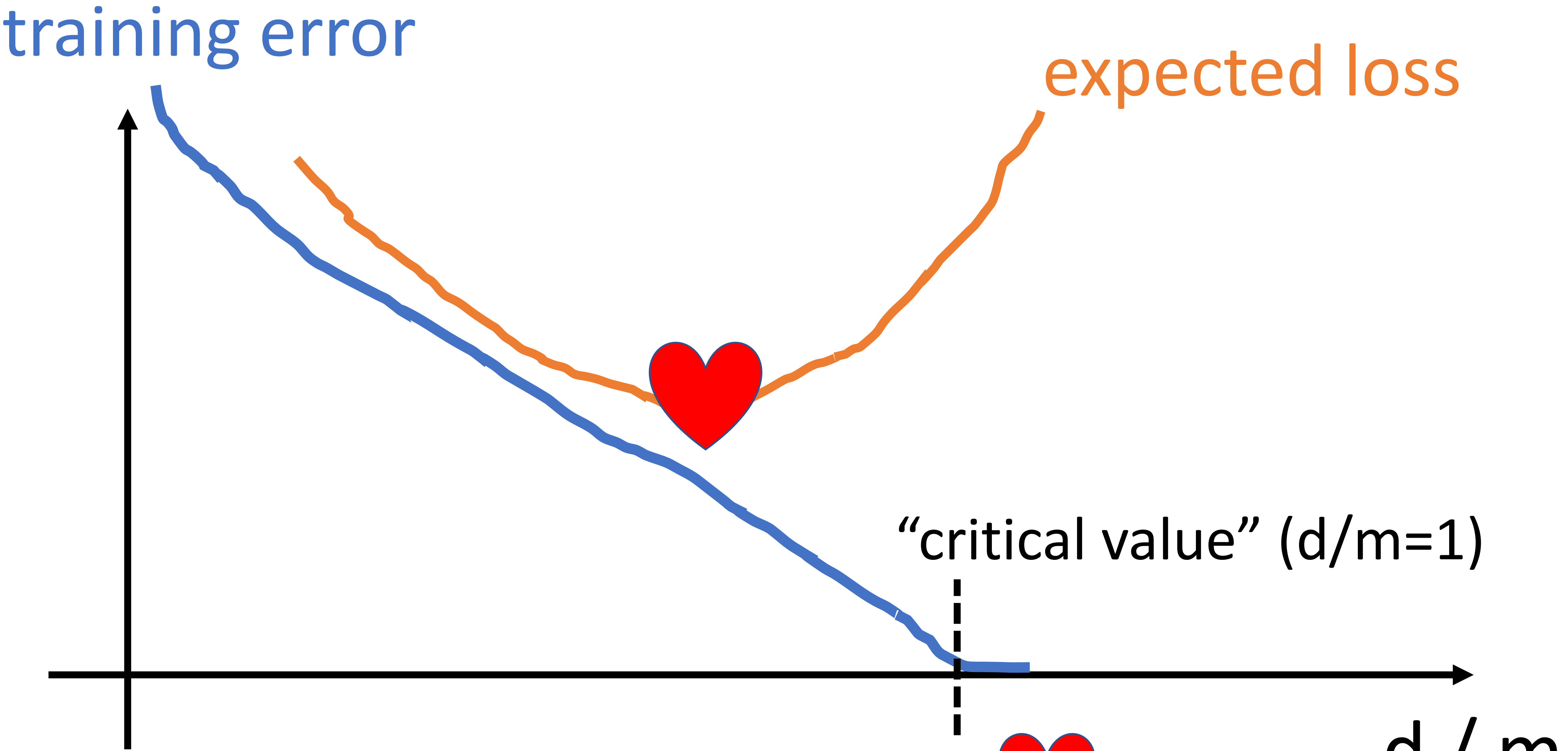


m=3, degree d=2 polynomial



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

how to bring d/m below critical value?

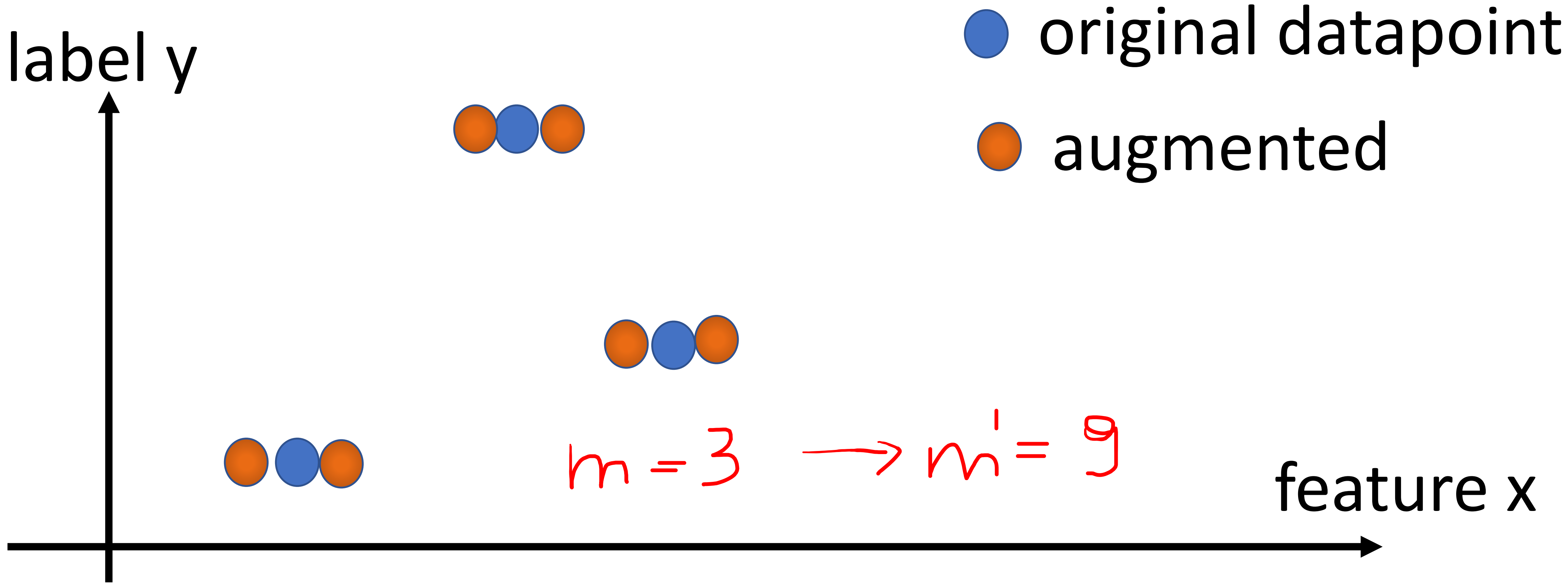
- increase m by using more training data
- decrease d by using smaller hypothesis space

how to bring d/m below critical value?

- increase m by using more training data
- decrease d by using smaller hypothesis space

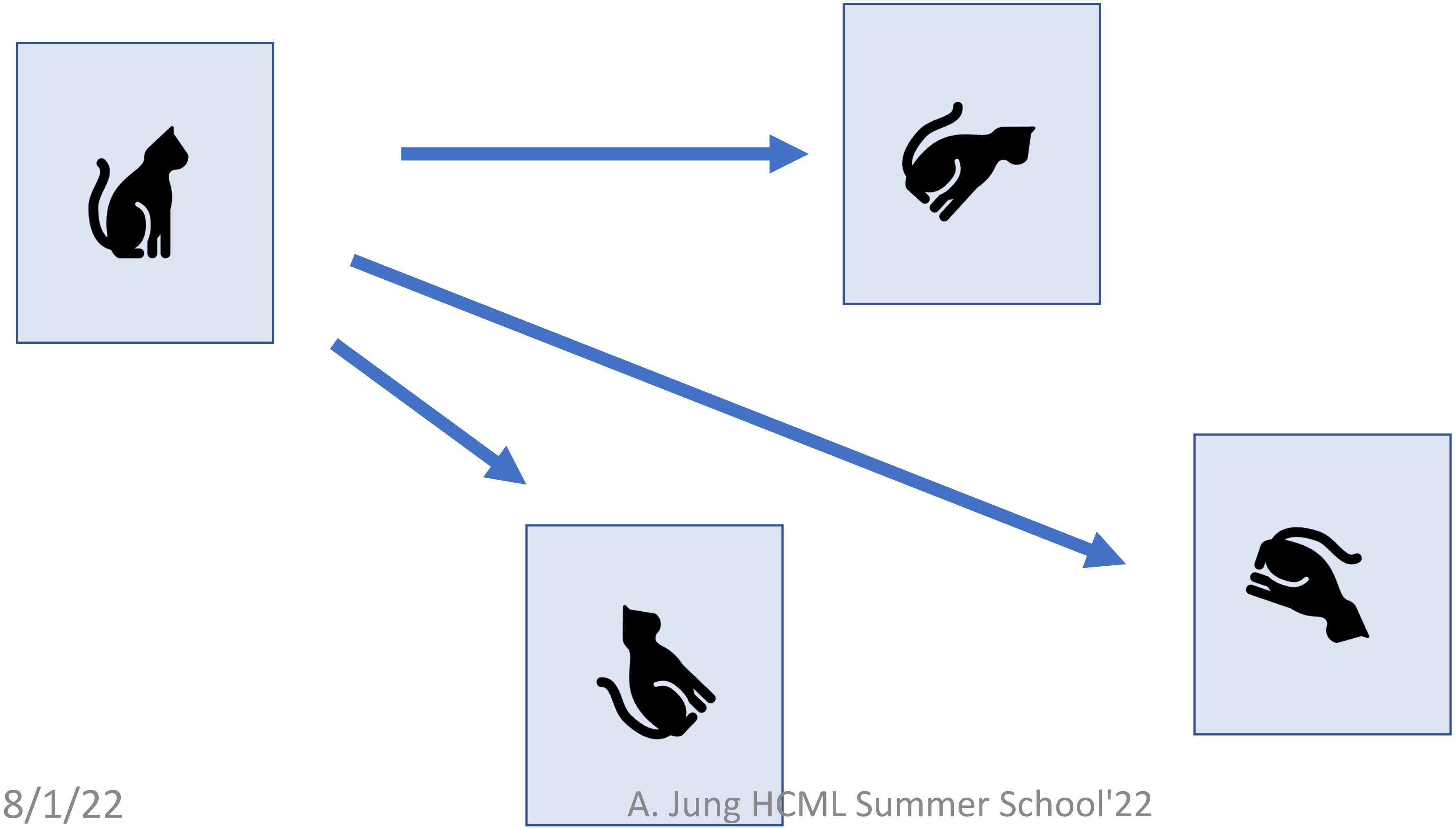
Data Augmentation

add a bit of noise to features

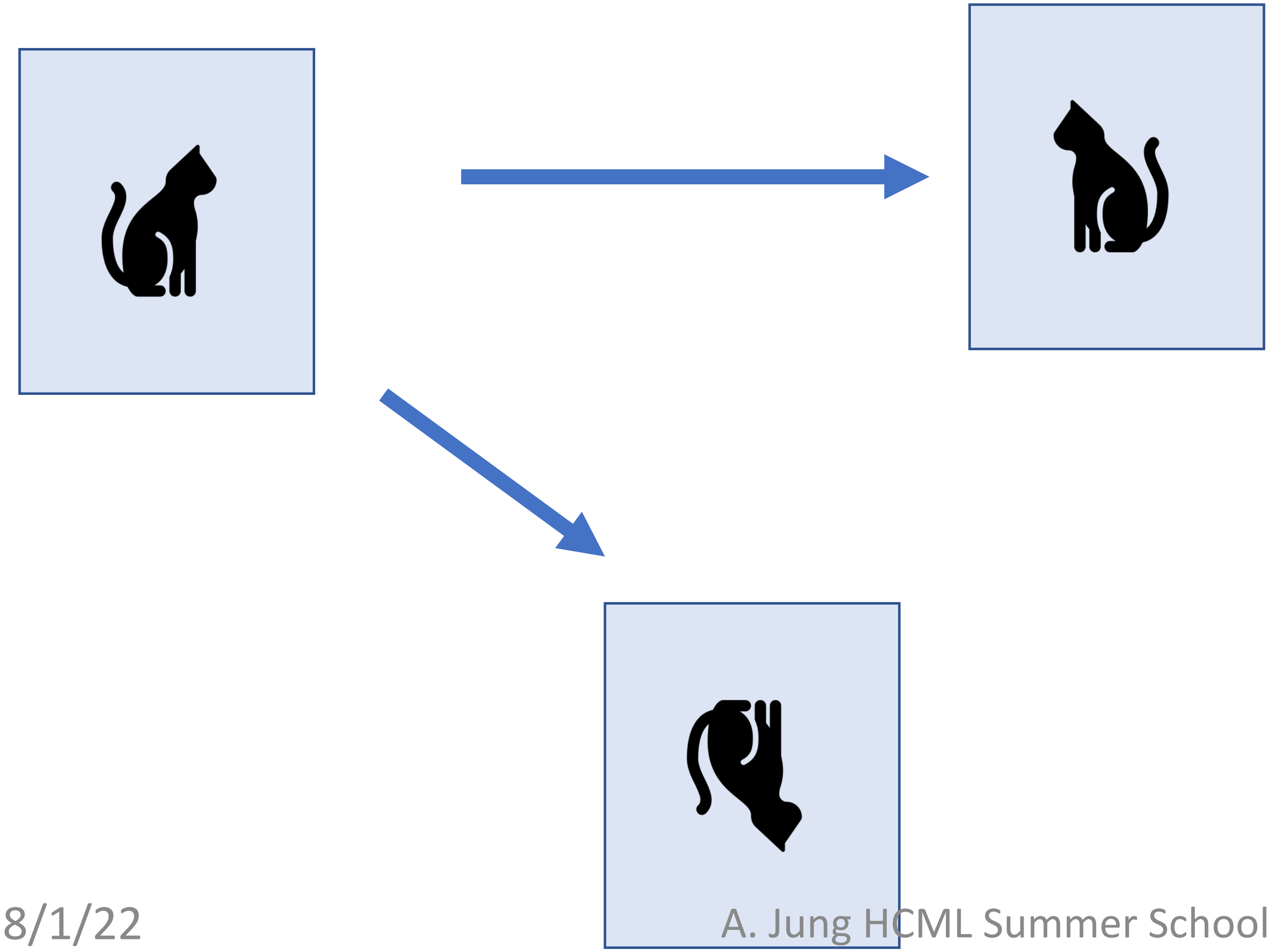


we have increased the dataset by factor 3 !

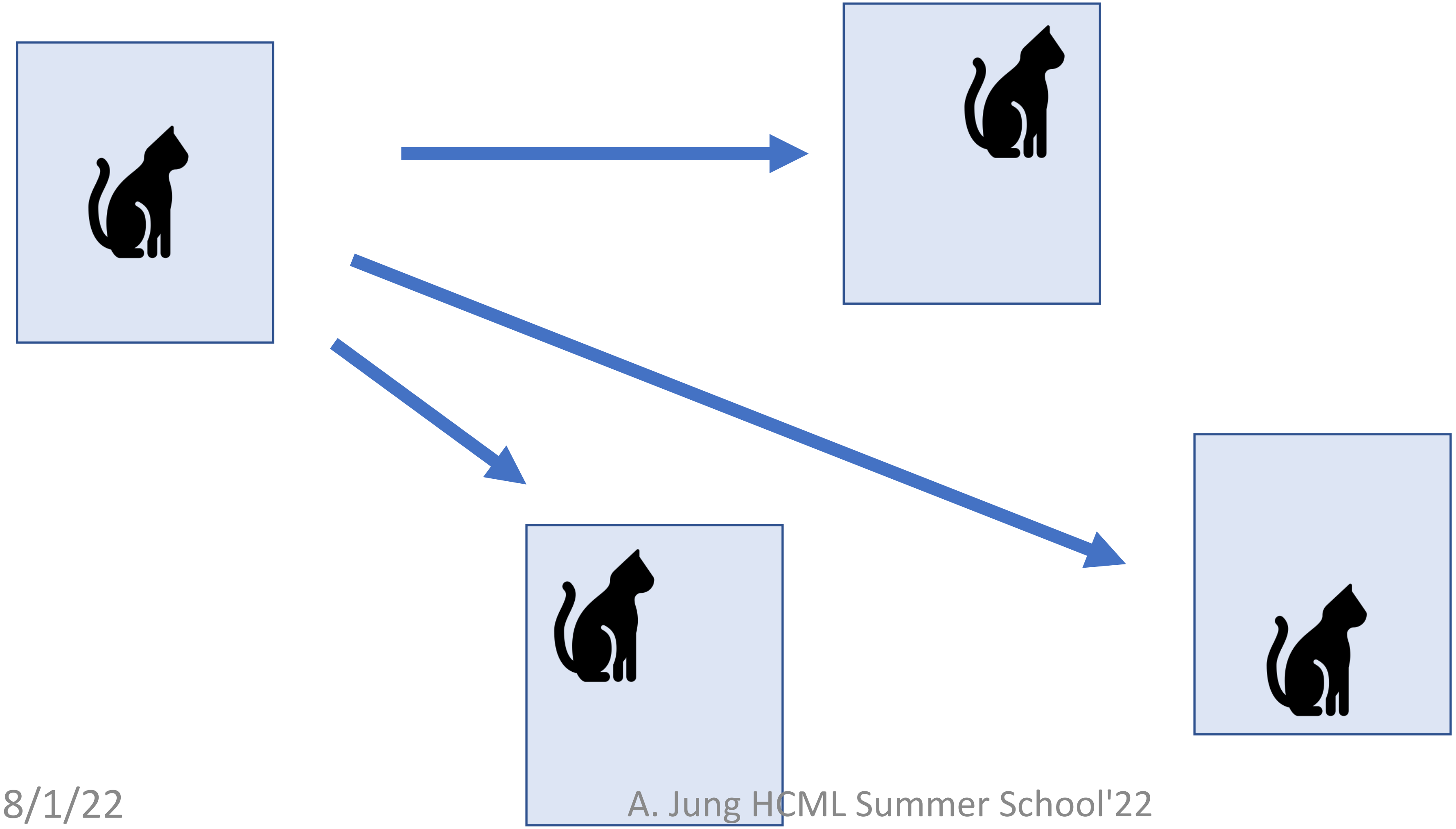
rotated cat image is still cat image



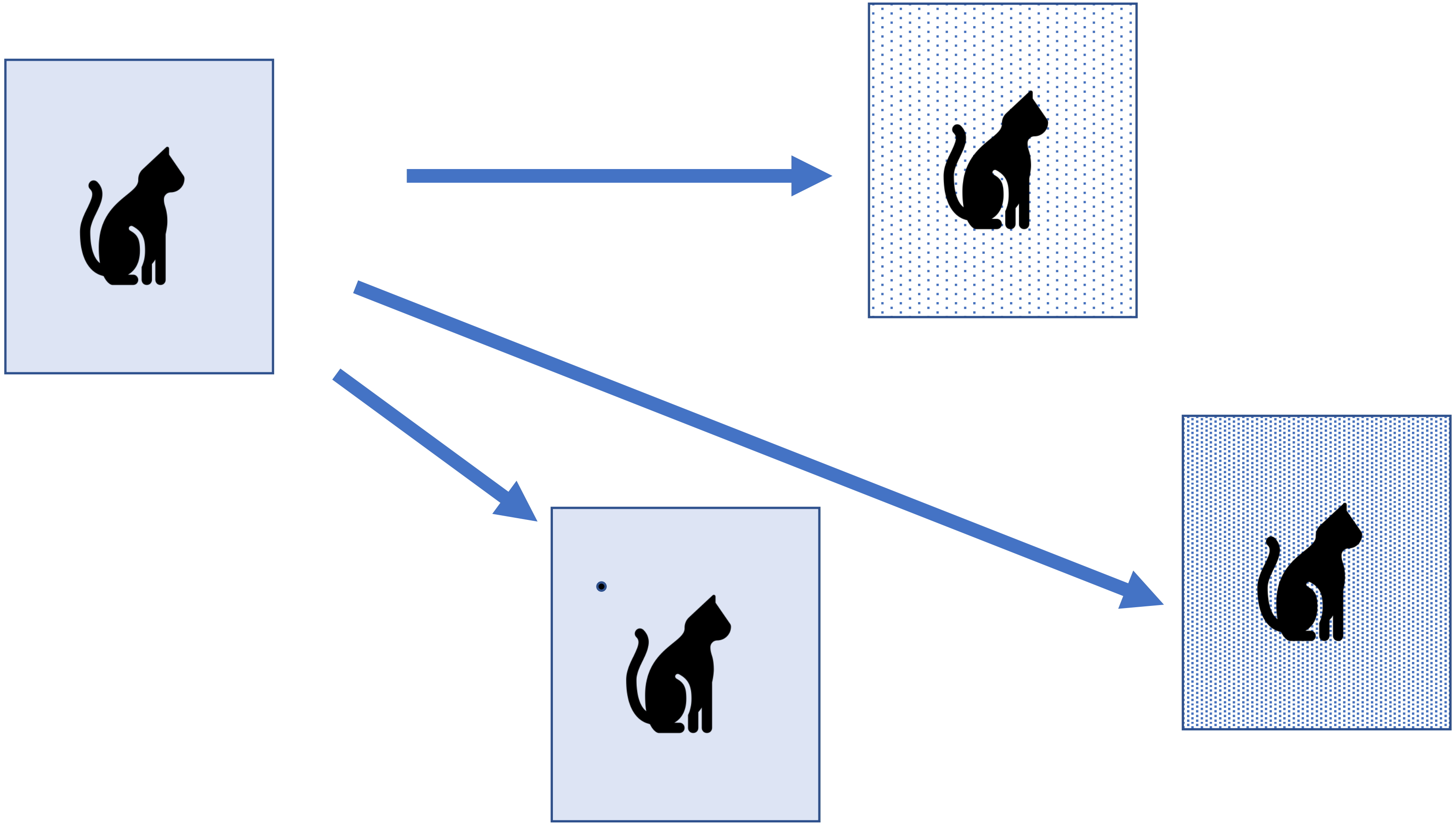
flipped cat image is still cat image



shifted cat image is still cat image



noisy cat image is still cat image



how to bring d/m below critical value?

- increase m by using more training data
- decrease d by using smaller hypothesis space

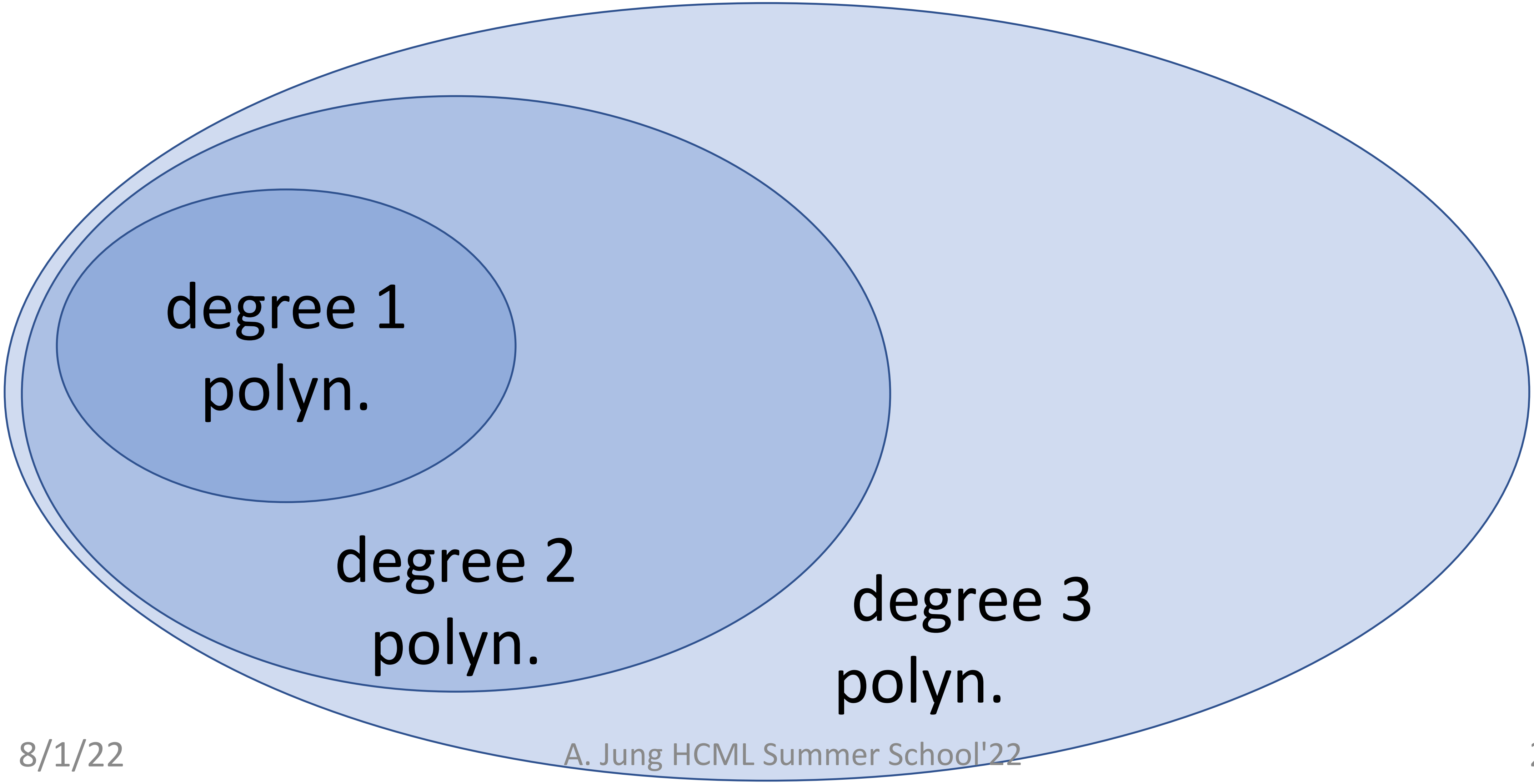
replace original ERM

$$\min_{h \epsilon \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} L\big((x^{(i)}, y^{(i)}), h\big)$$

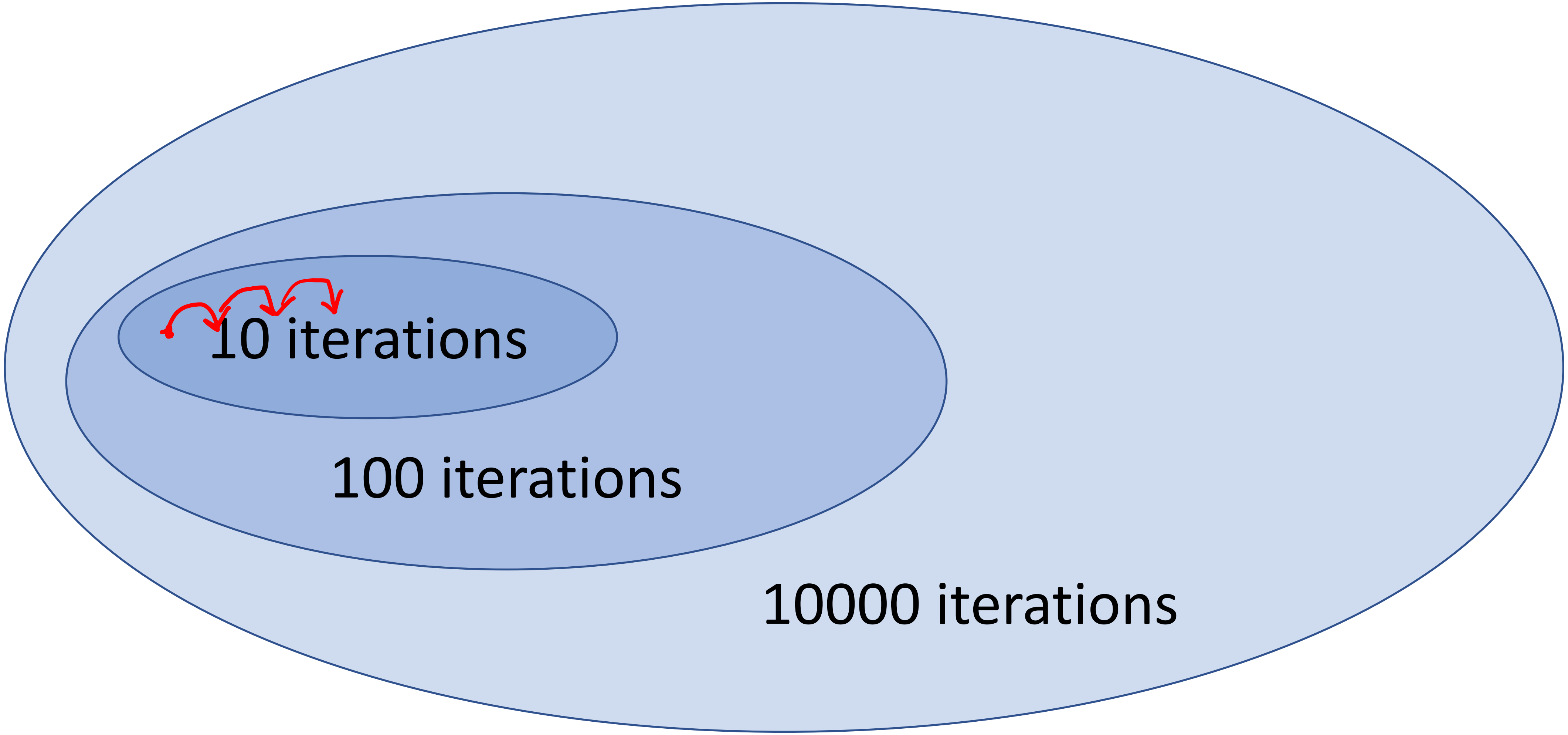
with ERM on smaller $\widehat{\mathcal{H}} \subset \mathcal{H}$

$$\min_{h \epsilon \widehat{\mathcal{H}}} \frac{1}{m} \sum_{i=1}^{m} L\big((x^{(i)}, y^{(i)}), h\big)$$

Nested Models



Prune Hypospace by Early Stopping



Soft Model Pruning via Regularization

Regularized ERM

learn hypothesis h out of model (hypospace) $\mathcal{H}$ by minimizing

train $\sum_{i=1}^{n} w_i^2$

min $h \in \mathcal{H}$

$$\underbrace{\frac{1}{m}\sum_{i=1}^{m} L\big((x^{(i)},y^{(i)}),h\big)}_{\text{average loss on training set (empirical risk of h)}} + \underbrace{\lambda \mathcal{R}(h)}_{\text{loss increase for datapoints outside training set}}$$

Regularized Linear Regression

- squared error loss
- linear hypothesis map $h^{(w)}(x) = w^T x = w_1 x_1 + \dots + w_n x_n$

$$\min_{w} \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - w^T x^{(i)}\right)^2 + \lambda \mathcal{R}(w)$$

- ridge regression uses $\mathcal{R}(w) = \|w\|_2^2 = w_1^2 + \dots + w_n^2$

sparse

- Lasso uses $\mathcal{R}(w) = \|w\|_1 = |w_1| + \dots + |w_n|$

Regularization = Implicit Pruning!

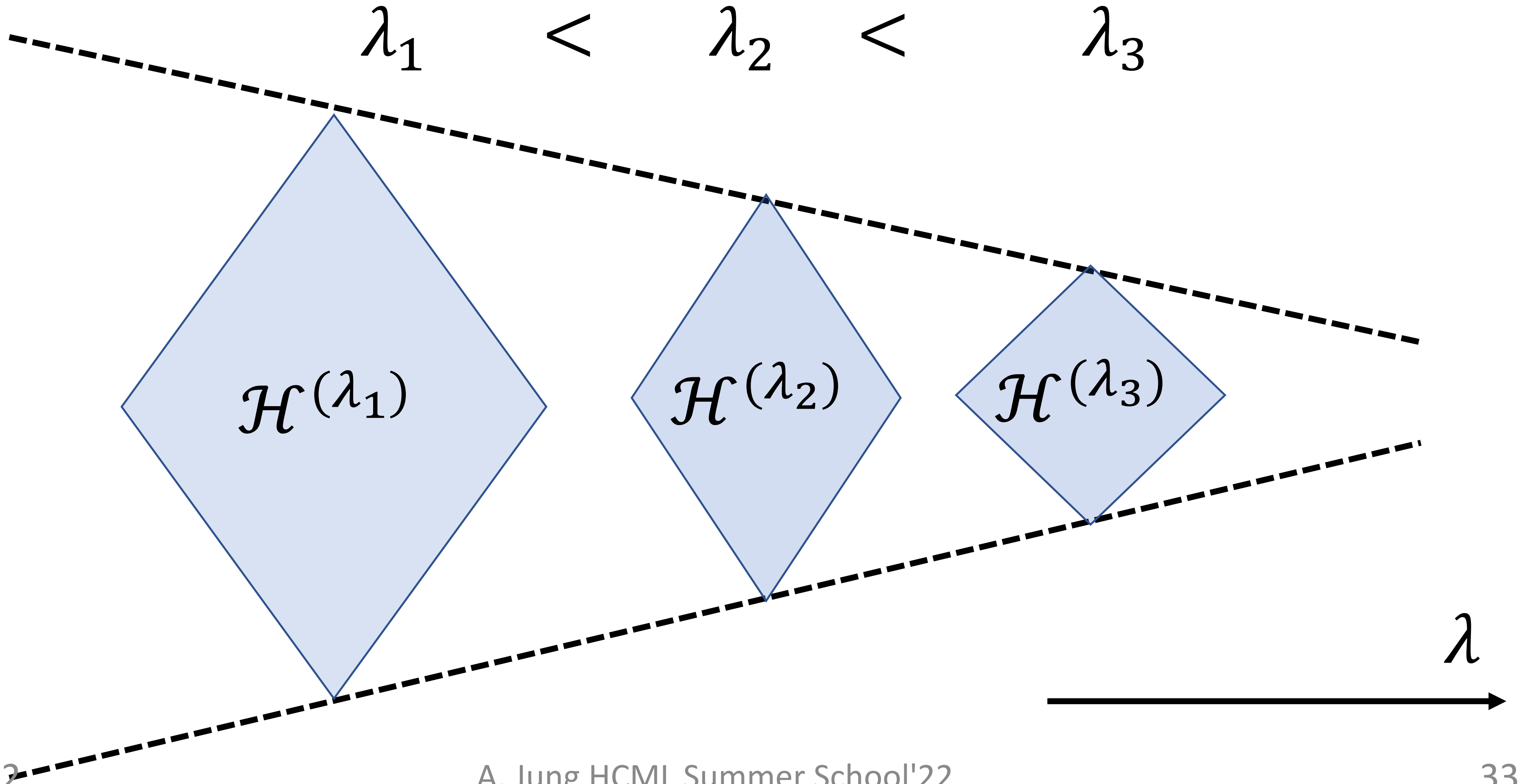
$$\min_{h \epsilon \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} L\left(\left(x^{(i)}, y^{(i)}\right), h\right) + \boxed{\cancel{\lambda} R(h)}$$

equivalent to

$$\min_{h \epsilon \mathcal{H}^{(\lambda)}} \frac{1}{m} \sum_{i=1}^{m} L\left(\left(x^{(i)}, y^{(i)}\right), h\right)$$

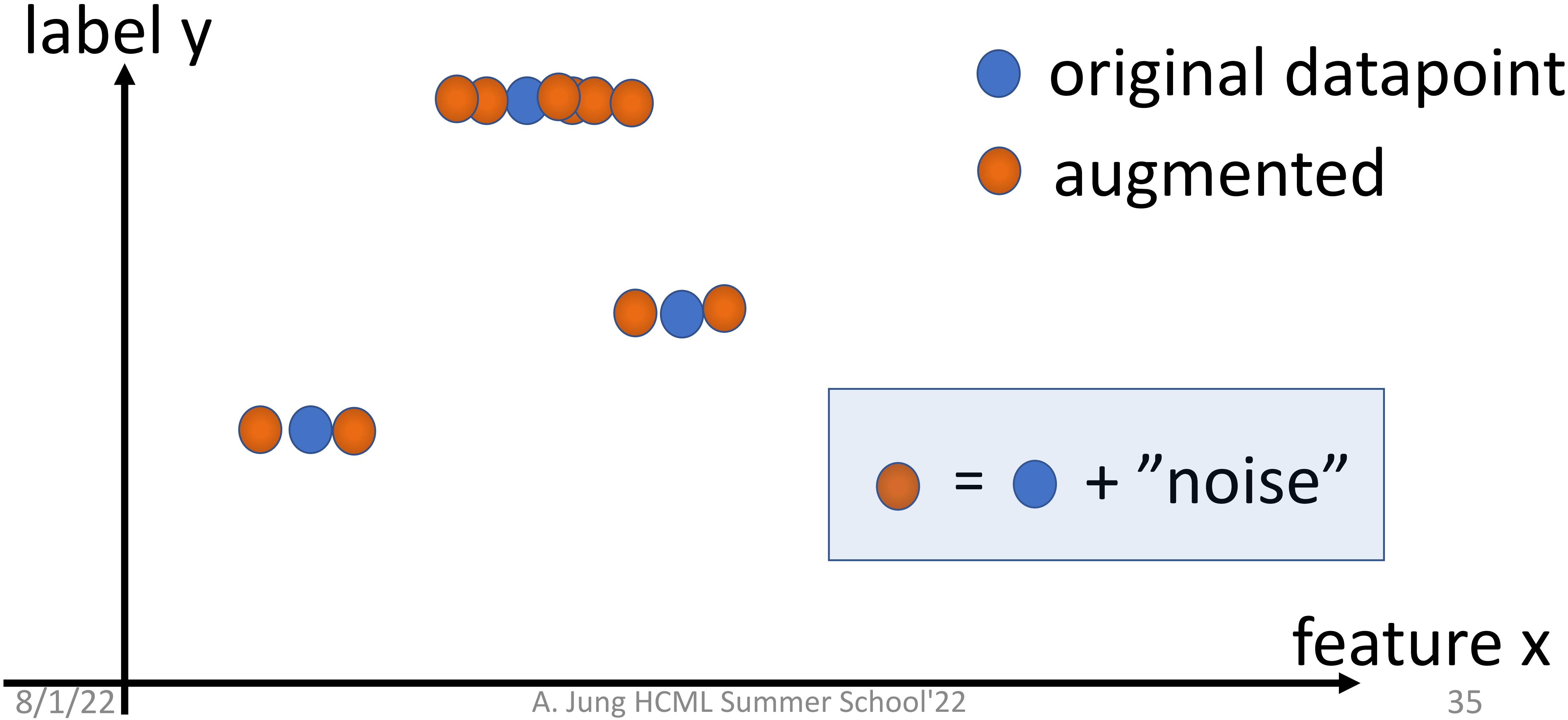
with pruned model $\mathcal{H}^{(\lambda)} \subset \mathcal{H}$

Regularization = “Soft” Model Selection

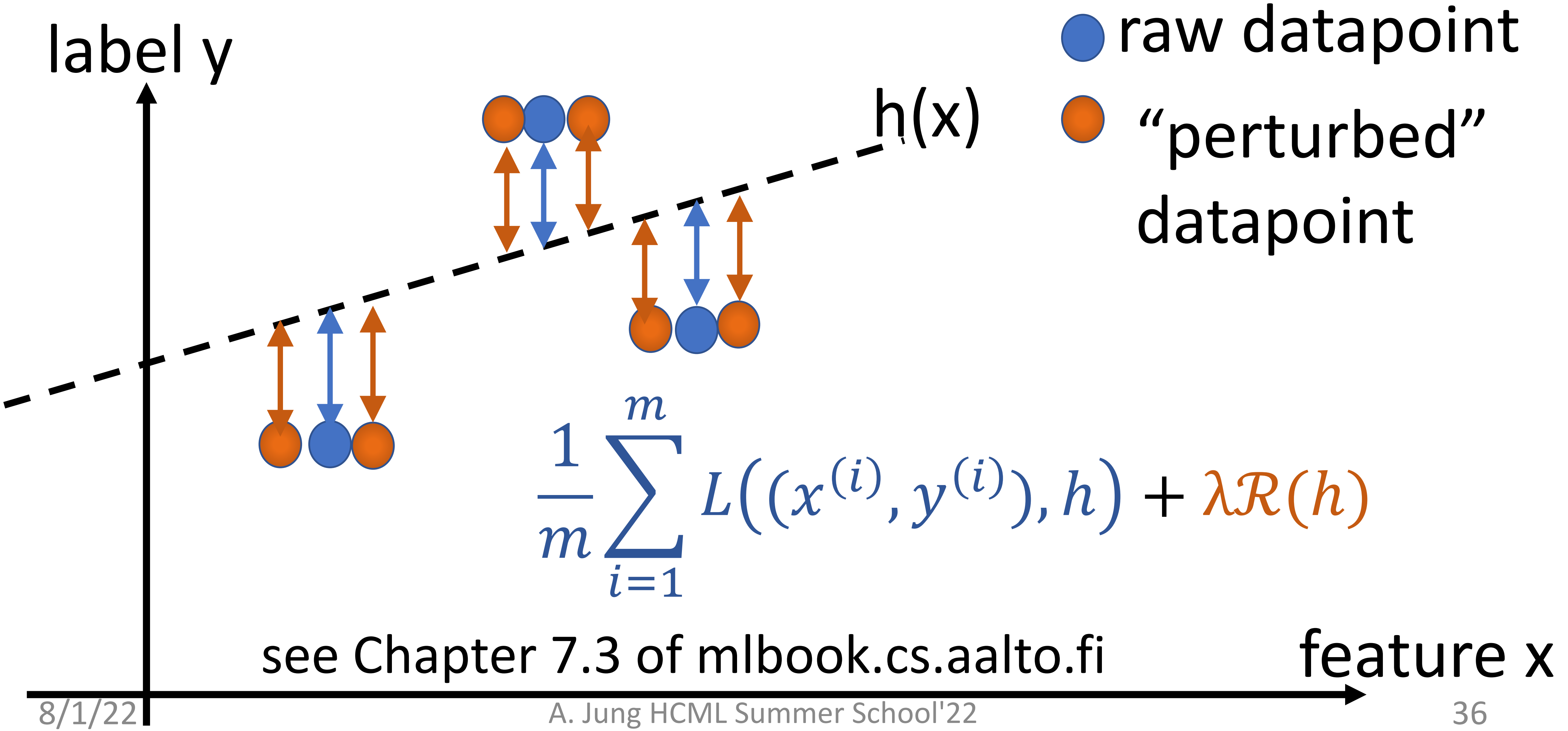


Regularization does implicit Data Augmentation

augment with (infinitely many) realizations of RV!



Regularization =Implicit Data Aug.



$$\frac{1}{m}\sum_{i=1}^{m} L\big((x^{(i)}, y^{(i)}), h\big) + \lambda\mathcal{R}(h)$$

see Chapter 7.3 of mlbook.cs.aalto.fi

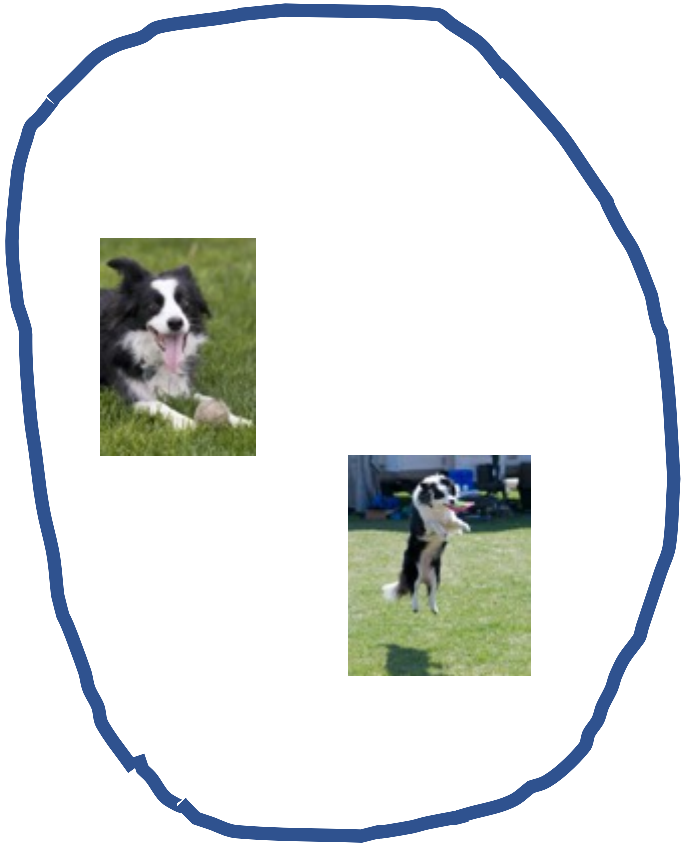
To sum up,

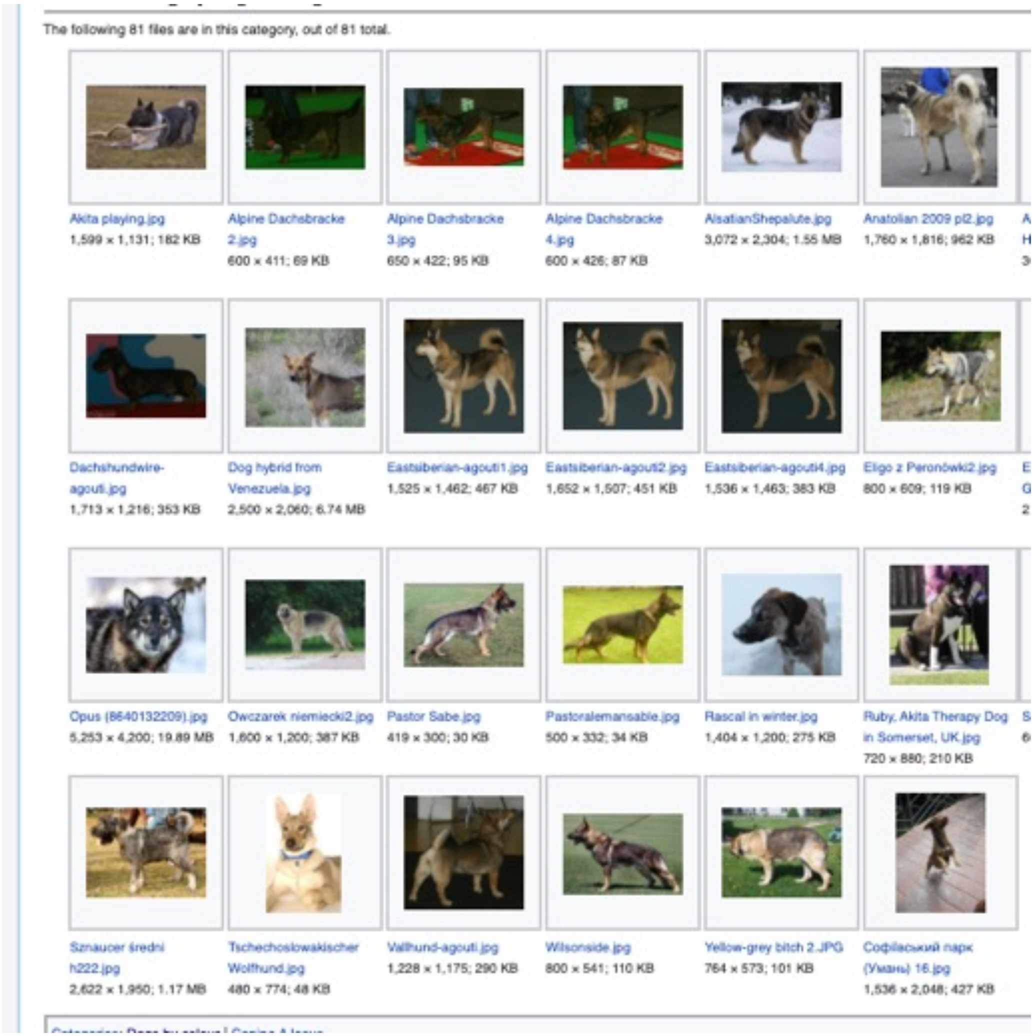
cont @ 15:15

- large ratio d/m leads to overfitting
- reduce d by using smaller model ("pruning")
- increase m by using more data points
- regularization is a soft model pruning
- regularization does implicit data augmentation

Transfer Learning via Regularization

- Problem I: classify image as “shows border collie” vs. “not”
- Problem II: classify image as “shows a dog” vs. “not”
- ML Problem I is our main interest
- only little training data $\mathcal{D}^{(1)}$ for Problem I
- much more labeled data $\mathcal{D}^{(2)}$ for Problem II
- pre-train a hypothesis on $\mathcal{D}^{(2)}$, fine-tune on $\mathcal{D}^{(1)}$





$\mathcal{D}^{(1)}$

learn h by fine-tuning $\hat{h}$

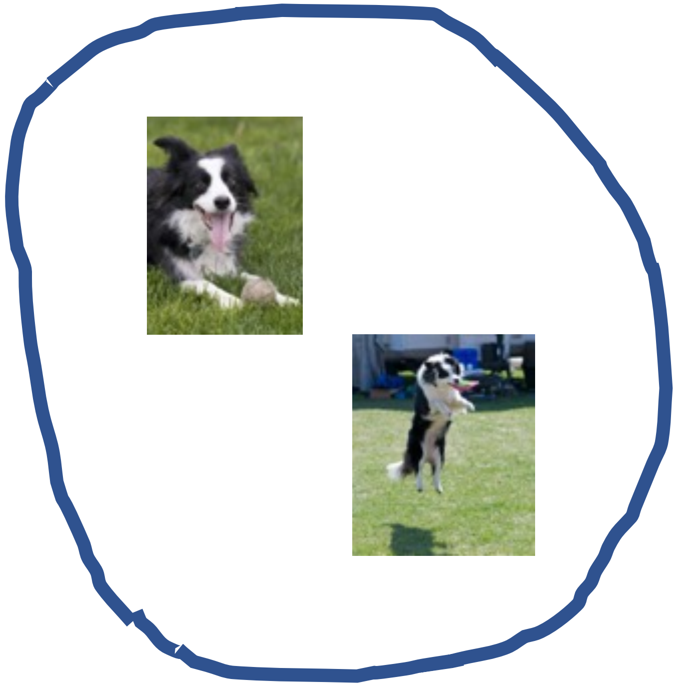
$\mathcal{D}^{(2)}$

pre-train hypothesis $\hat{h}$

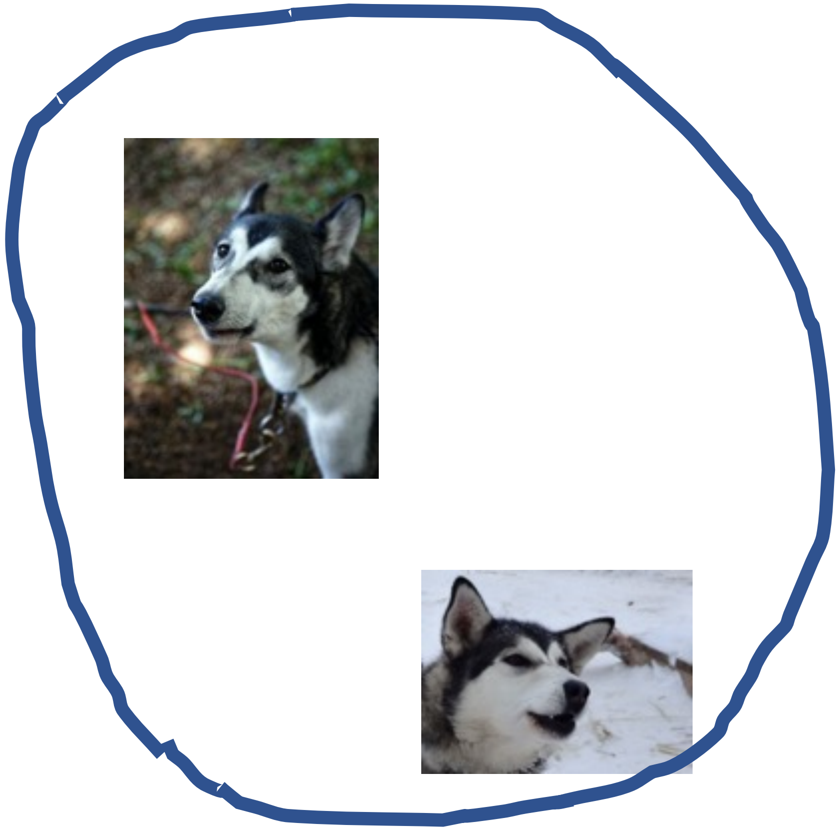
$$\underbrace{\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} L\left(\left(x^{(i)}, y^{(i)}\right), h\right)}_{\text{fine tuning on } \mathcal{D}^{(1)}} + \underbrace{\lambda d\left(h, \hat{h}\right)}_{\text{distance to hypothesis } \hat{h} \text{ which is pre-trained on } \mathcal{D}^{(2)}}$$

Multi-Task Learning via Regularization

- Problem I: classify image as “shows border colly” vs. “not”
- Problem II: classify image as “shows husky” vs. “not”
- training data $\mathcal{D}^{(1)}$ for Problem I and $\mathcal{D}^{(2)}$ for Problem II
- jointly learn hypothesis $h^{(1)}$ on $\mathcal{D}^{(1)}$ and $h^{(2)}$ on $\mathcal{D}^{(2)}$
- require $h^{(1)}$ to be “similar” to $h^{(2)}$



$\mathcal{D}^{(1)}$



$\mathcal{D}^{(2)}$

jointly learn similar
$h^{(1)}$ and $h^{(2)}$ for each dataset

training error of $h^{(1)}$

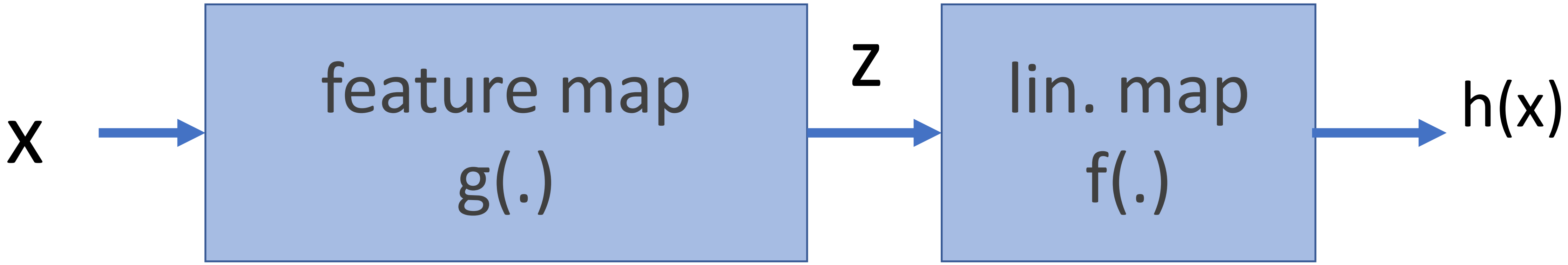
training error of $h^{(2)}$

$$\min_{h^{(1)},h^{(2)}} \hat{L}\left(h^{(1)}\middle|\mathcal{D}^{(1)}\right)+\hat{L}\left(h^{(2)}\middle|\mathcal{D}^{(2)}\right) + \lambda d\left(h^{(1)},h^{(2)}\right)$$

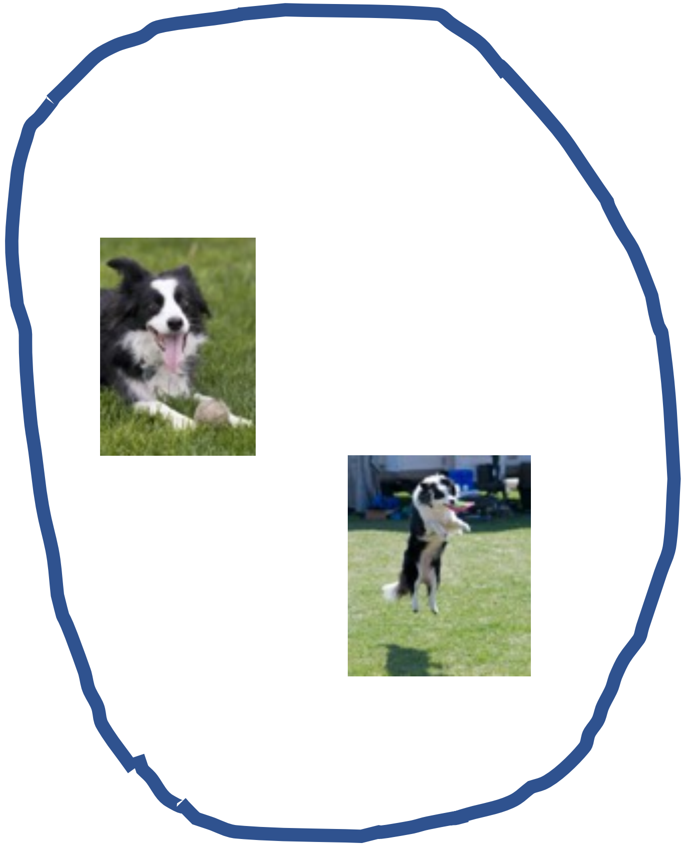
“distance” between $h^{(1)}$ and $h^{(2)}$

Semi-Supervised Learning via Regularization

- classify image as “shows border colly” vs. “not”
- small labeled dataset $\mathcal{D}^{(1)}$
- massive image database $\mathcal{D}^{(2)}$ with unlabeled images
- train hypothesis h(.) on $\mathcal{D}^{(1)}$ with following structure:



“chain” or “pipeline”



$\mathcal{D}^{(1)}$

learn linear classifier f(.)

$\mathcal{D}^{(2)}$

learn feature map g(.)

$$\underbrace{\min_{h \epsilon \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} L\left(\left(x^{(i)}, y^{(i)}\right), h\right)}_{} + \lambda \underbrace{\hat{L}\left(g \middle| \mathcal{D}^{(2)}\right)}_{}$$

use training error
to fine tune h(.)

learn feature map g(.)
using large unlabeled
database $\mathcal{D}^{(2)}$

To Sum Up

- ML works well if m/d > 1
- increase data size m by data augmentation
- decease model size d by regularization
- adding reg. term = data augmentation/soft model-pruning
- special cases of reg.: transfer-, multi-task- and semi-supervised learning