

# 10-301/601: Introduction to Machine Learning

## Lecture 5 – Model Selection

Henry Chai

5/21/24

# Front Matter

- Announcements:
  - HW2 released on 5/16, due 5/23 at 11:59 PM
- Recommended Readings:
  - Daumé III, [Chapter 2: Limits of Learning](#)

## Recall: Setting $k$

- When  $k = 1$ :
  - many, complicated decision boundaries
  - may overfit
- When  $k = N$ :
  - no decision boundaries; always predicts the most common label in the training data
  - may underfit
- $k$  controls the complexity of the hypothesis set  $\implies k$  affects how well the learned hypothesis will generalize

## Recall: Setting $k$

- Theorem:
  - If  $k$  is some function of  $N$  s.t.  $k(N) \rightarrow \infty$  and  $\frac{k(N)}{N} \rightarrow 0$  as  $N \rightarrow \infty$  ...
  - ... then (under certain assumptions) the true error of a  $k$ NN model  $\rightarrow$  the Bayes error rate
- Practical heuristics:
  - $k = \lfloor \sqrt{N} \rfloor$
  - $k = 3$
- This is a question of **model selection**: each value of  $k$  corresponds to a different “model”

# Model Selection

- A **model** is a (typically infinite) set of classifiers that a learning algorithm searches through to find the best one (the "hypothesis space")
- **Model parameters** are the numeric values or structure that are selected by the learning algorithm
- **Hyperparameters** are the tunable aspects of the model that are not selected by the learning algorithm

## Example: Decision Trees

- Model = set of all possible trees, potentially narrowed down according to the hyperparameters (see below)
- Model parameters = structure of a specific tree e.g., splits, split order, predictions at leaf nodes,
- Hyperparameters = splitting criterion, max-depth, tie-breaking procedures, etc...

# Model Selection

- A **model** is a (typically infinite) set of classifiers that a learning algorithm searches through to find the best one (the “hypothesis space”)
- **Model parameters** are the numeric values or structure that are selected by the learning algorithm
- **Hyperparameters** are the tunable aspects of the model that are not selected by the learning algorithm

## Example: $k$ NN

- Model = set of all possible nearest neighbors classifiers
- Model parameters = none!  $k$ NN is a “non-parametric model”
- Hyperparameters =  $k$

# Model Selection with Test Sets

- Given  $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$ , suppose we have multiple candidate models:

$$\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_M$$

- Learn a classifier from each model using only  $\mathcal{D}_{train}$ :

$$h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2, \dots, h_M \in \mathcal{H}_M$$

- Evaluate each one using  $\mathcal{D}_{test}$  and choose the one with lowest test error:

$$\hat{m} = \operatorname{argmin}_{m \in \{1, \dots, M\}} \operatorname{err}(h_m, \mathcal{D}_{test})$$

# Model Selection with Test Sets?

- Given  $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{test}$ , suppose we have multiple candidate models:

$$\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_M$$

- Learn a classifier from each model using only  $\mathcal{D}_{train}$ :

$$h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2, \dots, h_M \in \mathcal{H}_M$$

- Evaluate each one using  $\mathcal{D}_{test}$  and choose the one with lowest test error:

$$\hat{m} = \operatorname{argmin}_{m \in \{1, \dots, M\}} \operatorname{err}(h_m, \mathcal{D}_{test})$$

- Is  $\operatorname{err}(h_{\hat{m}}, \mathcal{D}_{test})$  a good estimate of  $\operatorname{err}(h_{\hat{m}})$ ?



# Model Selection with Validation Sets

- Given  $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$ , suppose we have multiple candidate models:

$$\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_M$$

- Learn a classifier from each model using only  $\mathcal{D}_{train}$ :

$$h_1 \in \mathcal{H}_1, h_2 \in \mathcal{H}_2, \dots, h_M \in \mathcal{H}_M$$

- Evaluate each one using  $\mathcal{D}_{val}$  and choose the one with lowest *validation* error:

$$\hat{m} = \operatorname{argmin}_{m \in \{1, \dots, M\}} \operatorname{err}(h_m, \mathcal{D}_{val})$$

- Now  $\operatorname{err}(h_{\hat{m}}, \mathcal{D}_{test})$  is a good estimate of  $\operatorname{err}(h_{\hat{m}})$ !

# Hyperparameter Optimization with Validation Sets

- Given  $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$ , suppose we have multiple candidate hyperparameter settings:

$$\theta_1, \theta_2, \dots, \theta_M$$

- Learn a classifier for each setting using only  $\mathcal{D}_{train}$ :

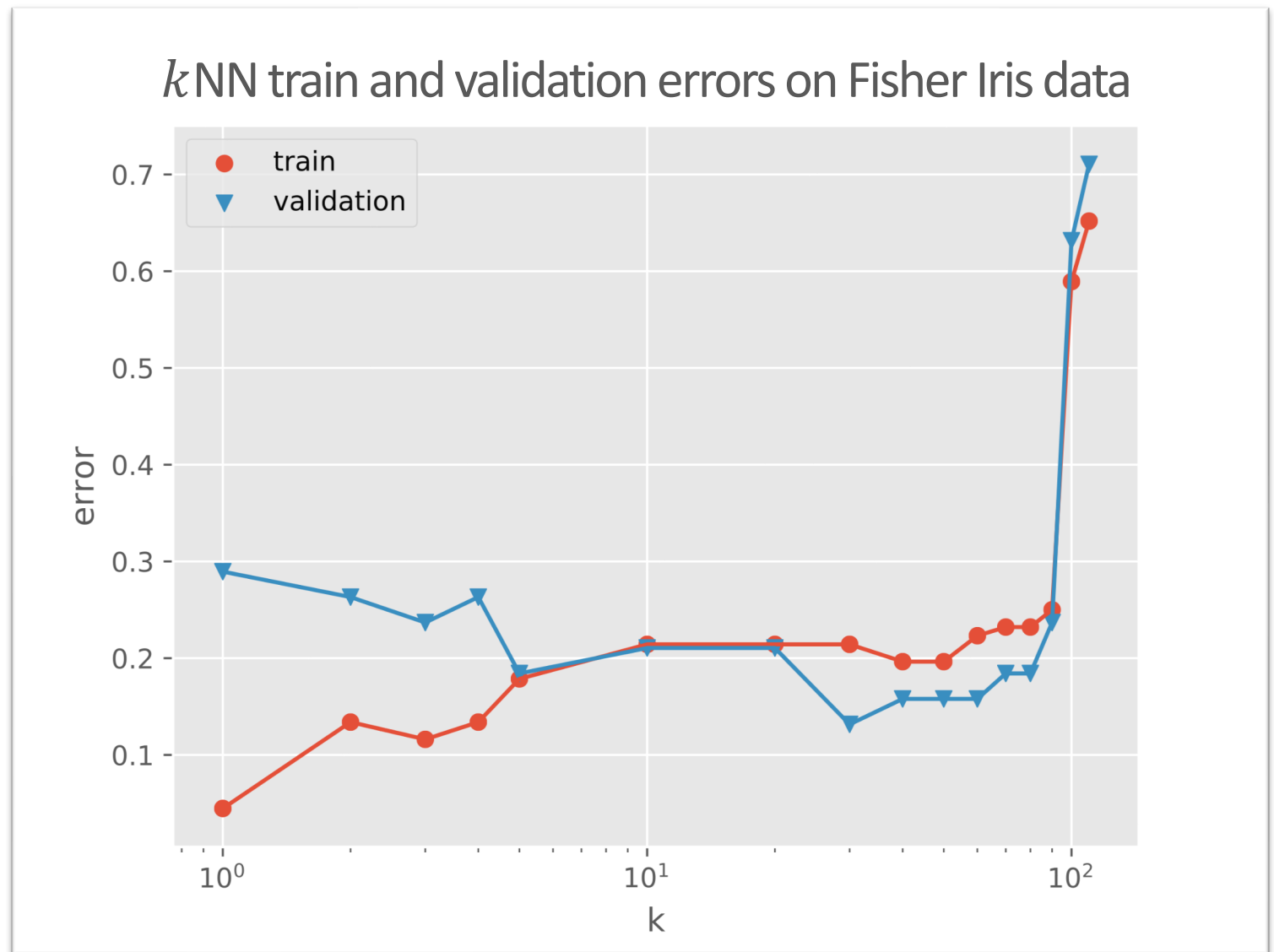
$$h_1, h_2, \dots, h_M$$

- Evaluate each one using  $\mathcal{D}_{val}$  and choose the one with lowest *validation* error:

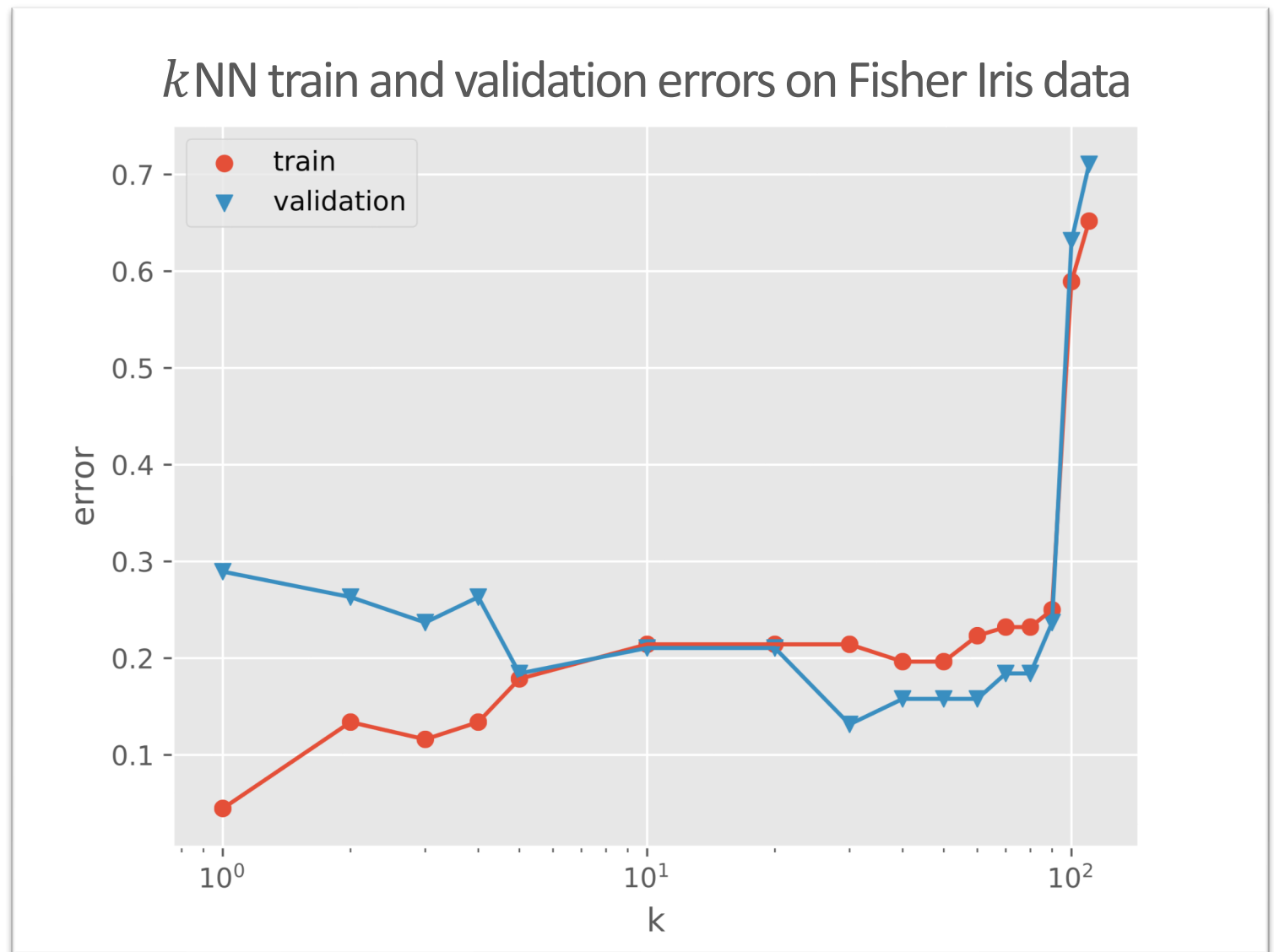
$$\hat{m} = \operatorname{argmin}_{m \in \{1, \dots, M\}} \operatorname{err}(h_m, \mathcal{D}_{val})$$

- Now  $\operatorname{err}(h_{\hat{m}}, \mathcal{D}_{test})$  is a good estimate of  $\operatorname{err}(h_{\hat{m}})$ !

# Setting $k$ for $k$ NN with Validation Sets



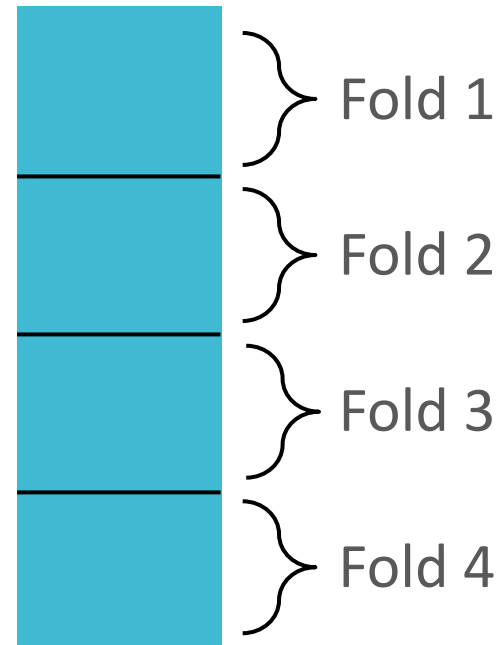
How should we partition our dataset?



# $K$ -fold cross-validation

- Given  $\mathcal{D}$ , split  $\mathcal{D}$  into  $K$  equally sized datasets or folds:  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$

- Use each one as a validation set once:



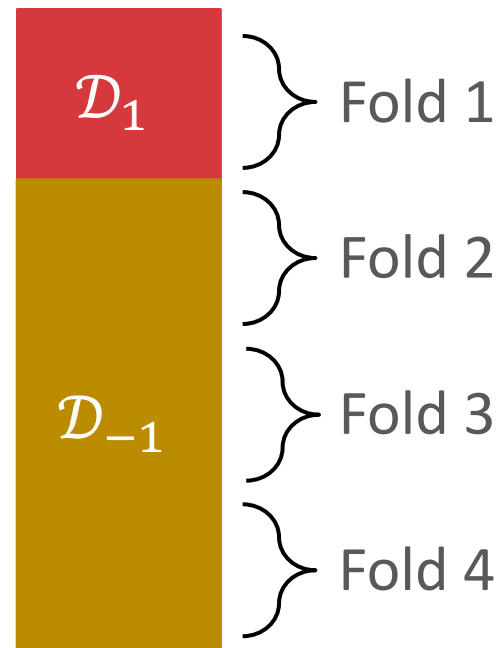
- Let  $h_{-i}$  be the classifier learned using  $\mathcal{D}_{-i} = \mathcal{D} \setminus \mathcal{D}_i$  (all folds other than  $\mathcal{D}_i$ ) and let  $e_i = \text{err}(h_{-i}, \mathcal{D}_i)$
- The  $K$ -fold cross validation error is

$$\text{err}_{cv_K} = \frac{1}{K} \sum_{i=1}^K e_i$$

# $K$ -fold cross-validation

- Given  $\mathcal{D}$ , split  $\mathcal{D}$  into  $K$  equally sized datasets or folds:  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$

- Use each one as a validation set once:



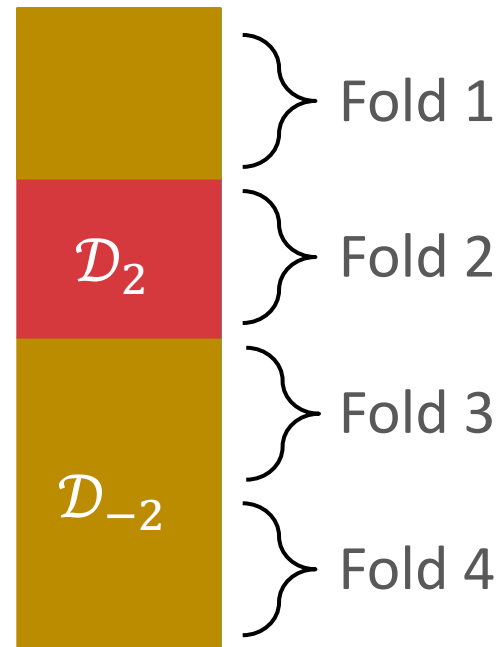
- Let  $h_{-i}$  be the classifier learned using  $\mathcal{D}_{-i} = \mathcal{D} \setminus \mathcal{D}_i$  (all folds other than  $\mathcal{D}_i$ ) and let  $e_i = \text{err}(h_{-i}, \mathcal{D}_i)$
- The  $K$ -fold cross validation error is

$$\text{err}_{cv_K} = \frac{1}{K} \sum_{i=1}^K e_i$$

# $K$ -fold cross-validation

- Given  $\mathcal{D}$ , split  $\mathcal{D}$  into  $K$  equally sized datasets or folds:  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$

- Use each one as a validation set once:



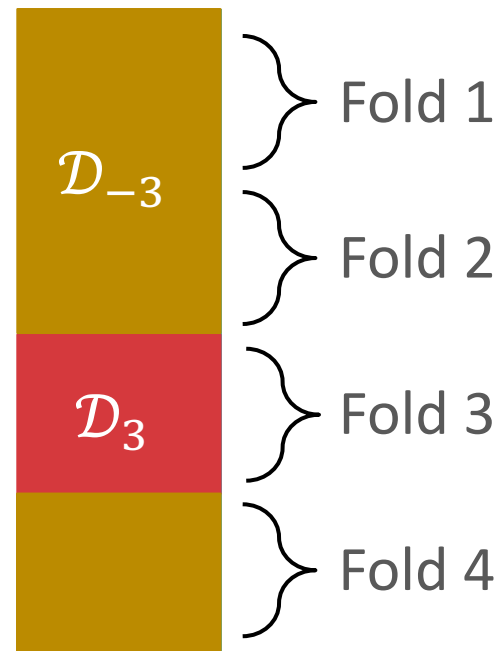
- Let  $h_{-i}$  be the classifier learned using  $\mathcal{D}_{-i} = \mathcal{D} \setminus \mathcal{D}_i$  (all folds other than  $\mathcal{D}_i$ ) and let  $e_i = \text{err}(h_{-i}, \mathcal{D}_i)$
- The  $K$ -fold cross validation error is

$$\text{err}_{cv_K} = \frac{1}{K} \sum_{i=1}^K e_i$$

# $K$ -fold cross-validation

- Given  $\mathcal{D}$ , split  $\mathcal{D}$  into  $K$  equally sized datasets or folds:  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$

- Use each one as a validation set once:



- Let  $h_{-i}$  be the classifier learned using  $\mathcal{D}_{-i} = \mathcal{D} \setminus \mathcal{D}_i$  (all folds other than  $\mathcal{D}_i$ ) and let  $e_i = \text{err}(h_{-i}, \mathcal{D}_i)$
- The  $K$ -fold cross validation error is

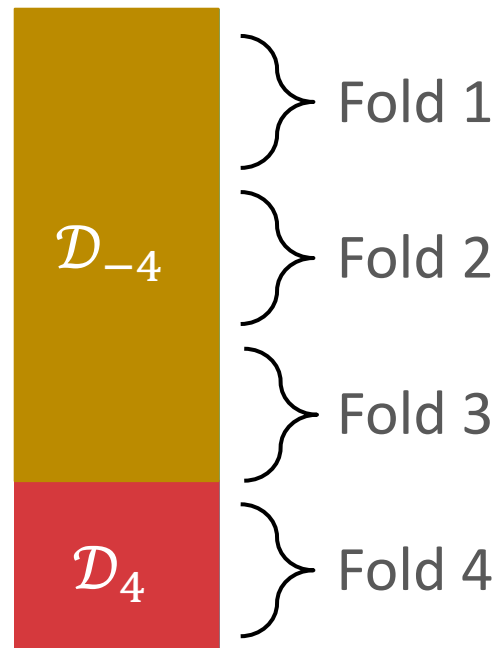
$$\text{err}_{cv_K} = \frac{1}{K} \sum_{i=1}^K e_i$$



# $K$ -fold cross-validation

- Given  $\mathcal{D}$ , split  $\mathcal{D}$  into  $K$  equally sized datasets or folds:  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$

- Use each one as a validation set once:



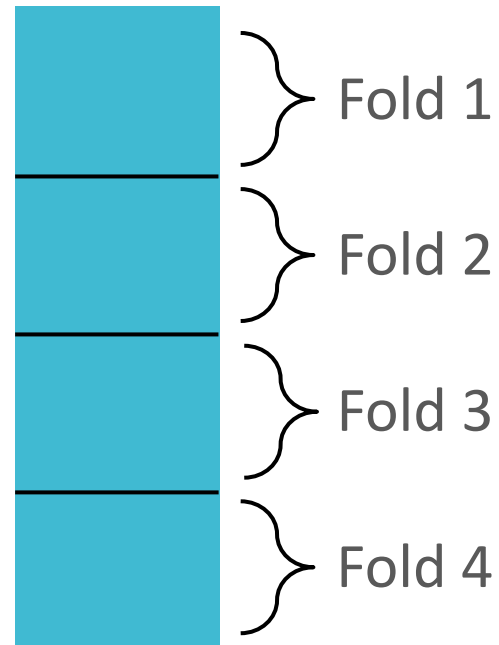
- Let  $h_{-i}$  be the classifier learned using  $\mathcal{D}_{-i} = \mathcal{D} \setminus \mathcal{D}_i$  (all folds other than  $\mathcal{D}_i$ ) and let  $e_i = \text{err}(h_{-i}, \mathcal{D}_i)$
- The  $K$ -fold cross validation error is

$$\text{err}_{cv_K} = \frac{1}{K} \sum_{i=1}^K e_i$$

# $K$ -fold cross-validation

- Given  $\mathcal{D}$ , split  $\mathcal{D}$  into  $K$  equally sized datasets or folds:  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_K$

- Use each one as a validation set once:



- Let  $h_{-i}$  be the classifier learned using  $\mathcal{D}_{-i} = \mathcal{D} \setminus \mathcal{D}_i$  (all folds other than  $\mathcal{D}_i$ ) and let  $e_i = \text{err}(h_{-i}, \mathcal{D}_i)$
- The  $K$ -fold cross validation error is

$$\text{err}_{cv_K} = \frac{1}{K} \sum_{i=1}^K e_i$$

- Special case when  $K = N$ : Leave-one-out cross-validation
- Choosing between  $m$  candidates requires training  $mK$  times

# Summary

	Input	Output
Training	<ul style="list-style-type: none"><li>• training dataset</li><li>• hyperparameters</li></ul>	<ul style="list-style-type: none"><li>• best model parameters</li></ul>
Hyperparameter Optimization	<ul style="list-style-type: none"><li>• training dataset</li><li>• validation dataset</li></ul>	<ul style="list-style-type: none"><li>• best hyperparameters</li></ul>
Cross-Validation	<ul style="list-style-type: none"><li>• training dataset</li><li>• validation dataset</li></ul>	<ul style="list-style-type: none"><li>• cross-validation error</li></ul>
Testing	<ul style="list-style-type: none"><li>• test dataset</li><li>• classifier</li></ul>	<ul style="list-style-type: none"><li>• test error</li></ul>

# Hyperparameter Optimization

- Given  $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$ , suppose we have multiple candidate hyperparameter settings:

$$\theta_1, \theta_2, \dots, \theta_M$$

- Learn a classifier for each setting using only  $\mathcal{D}_{train}$ :

$$h_1, h_2, \dots, h_M$$

- Evaluate each one using  $\mathcal{D}_{val}$  and choose the one with lowest *validation* error:

$$\hat{m} = \operatorname{argmin}_{m \in \{1, \dots, M\}} \operatorname{err}(h_m, \mathcal{D}_{val})$$

- Now  $\operatorname{err}(h_{\hat{m}}^+, \mathcal{D}_{test})$  is a good estimate of  $\operatorname{err}(h_{\hat{m}}^+)$ !

Pro tip: train your final model using *both* training and validation datasets

- Given  $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$ , suppose we have multiple candidate hyperparameter settings:

$$\theta_1, \theta_2, \dots, \theta_M$$

- Learn a classifier for each setting using only  $\mathcal{D}_{train}$ :

$$h_1, h_2, \dots, h_M$$

- Evaluate each one using  $\mathcal{D}_{val}$  and choose the one with lowest *validation* error:

$$\hat{m} = \operatorname{argmin}_{m \in \{1, \dots, M\}} \operatorname{err}(h_m, \mathcal{D}_{val})$$

- Train a new model on  $\mathcal{D}_{train} \cup \mathcal{D}_{val}$  using  $\theta_{\hat{m}}, h_{\hat{m}}^+$
- Now  $\operatorname{err}(h_{\hat{m}}^+, \mathcal{D}_{test})$  is a good estimate of  $\operatorname{err}(h_{\hat{m}}^+)$ !

# How do we pick hyperparameter settings to try?

- Given  $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$ , suppose we have multiple candidate hyperparameter settings:

$$\theta_1, \theta_2, \dots, \theta_M$$

- Learn a classifier for each setting using only  $\mathcal{D}_{train}$ :

$$h_1, h_2, \dots, h_M$$

- Evaluate each one using  $\mathcal{D}_{val}$  and choose the one with lowest *validation* error:

$$\hat{m} = \operatorname{argmin}_{m \in \{1, \dots, M\}} \operatorname{err}(h_m, \mathcal{D}_{val})$$

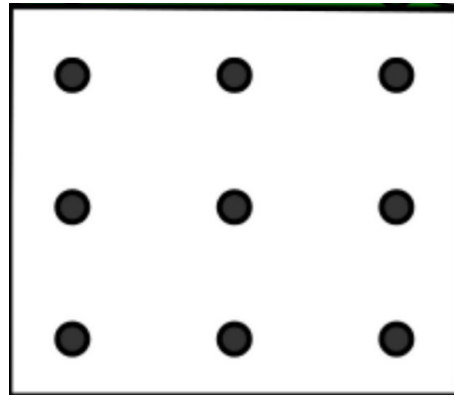
- Train a new model on  $\mathcal{D}_{train} \cup \mathcal{D}_{val}$  using  $\theta_{\hat{m}}, h_{\hat{m}}^+$
- Now  $\operatorname{err}(h_{\hat{m}}^+, \mathcal{D}_{test})$  is a good estimate of  $\operatorname{err}(h_{\hat{m}}^+)$ !

# General Methods for Hyperparameter Optimization

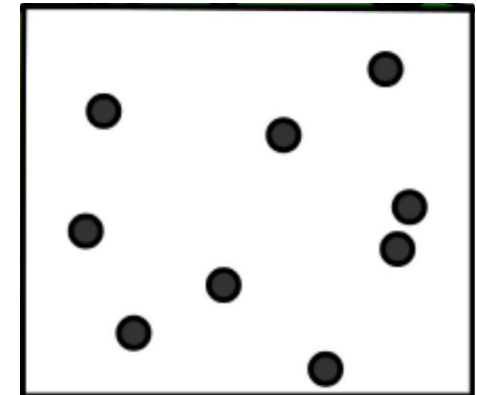
- Idea: set the hyperparameters to optimize some performance metric of the model
- Issue: if we have many hyperparameters that can all take on lots of different values, we might not be able to test all possible combinations
- Commonly used methods:
  - Grid search
  - Random search
  - Bayesian optimization (used by Google DeepMind to optimize the hyperparameters of AlphaGo: <https://arxiv.org/pdf/1812.06855v1.pdf>)
  - Evolutionary algorithms
  - Graduate-student descent

# Grid Search vs. Random Search (Bergstra and Bengio, 2012)

Grid Layout



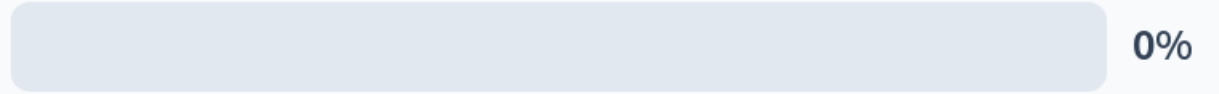
Random Layout



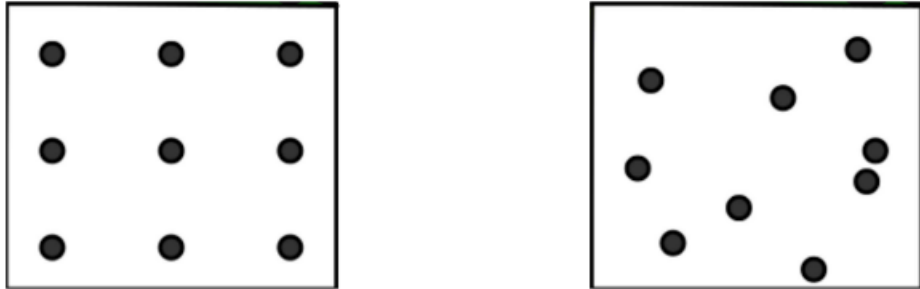
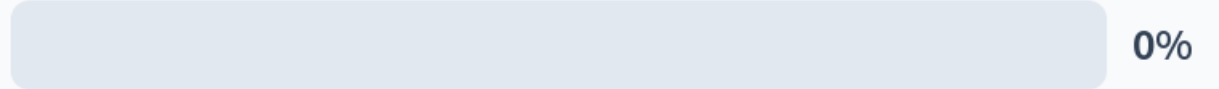


# In general, which hyperparameter optimization method do you think will perform better?

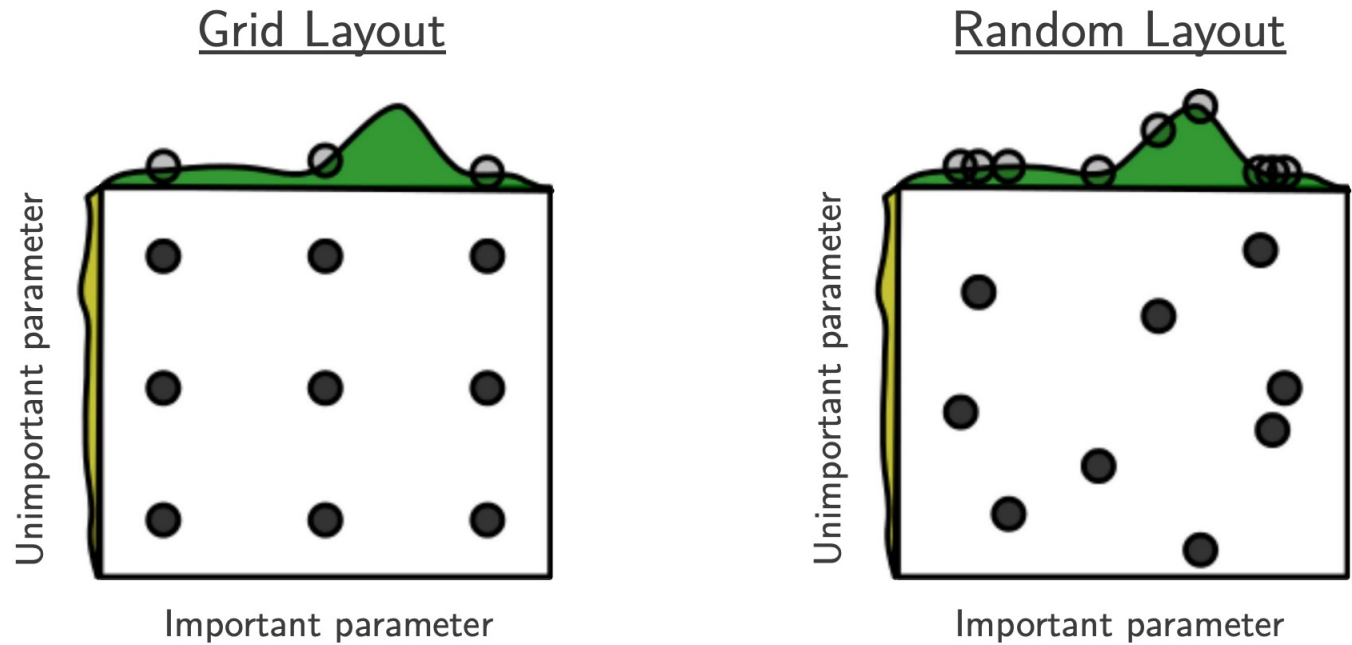
Grid Search



Random Search



# Grid Search vs. Random Search (Bergstra and Bengio, 2012)



Grid and random search of nine trials for optimizing a function  $f(x, y) = g(x) + h(y) \approx g(x)$  with *low effective dimensionality*. Above each square  $g(x)$  is shown in green, and left of each square  $h(y)$  is shown in yellow. With grid search, nine trials only test  $g(x)$  in three distinct places. With random search, all nine trials explore distinct values of  $g$ . This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.

# Key Takeaways

- Differences between training, validation and test datasets in the model selection process
- Cross-validation for model selection
- Relationship between training, hyperparameter optimization and model selection
- Grid search vs. random search for hyperparameter optimization