10-301/601: Introduction to Machine Learning Lecture 6 – Model Selection

Henry Chai

5/14/25

kNN: Pros and Cons

- Pros:
  - Intuitive / explainable
  - No training / retraining
  - Provably near-optimal in terms of true error rate
- Cons:
  - Computationally expensive
    - Always needs to store all data: $O(ND)$
    - Finding the k closest points in D dimensions: $O(ND + N\log(k))$
      - Can be sped up through clever use of data structures (trades off training and test costs)
      - Can be approximated using stochastic methods
  - Affected by feature scale

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 $\Longrightarrow$ k affects how well the learned hypothesis will generalize

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 kNN 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: kNN

- Model = set of all possible nearest neighbors classifiers
- Model parameters = none! kNN is a “non-parametric model”
- Hyperparameters = k, distance metric, tie-breaking, feature prioritization/scaling, outlier filtering

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:

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

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:

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

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

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:

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

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:

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

Setting k for kNN with Validation Sets

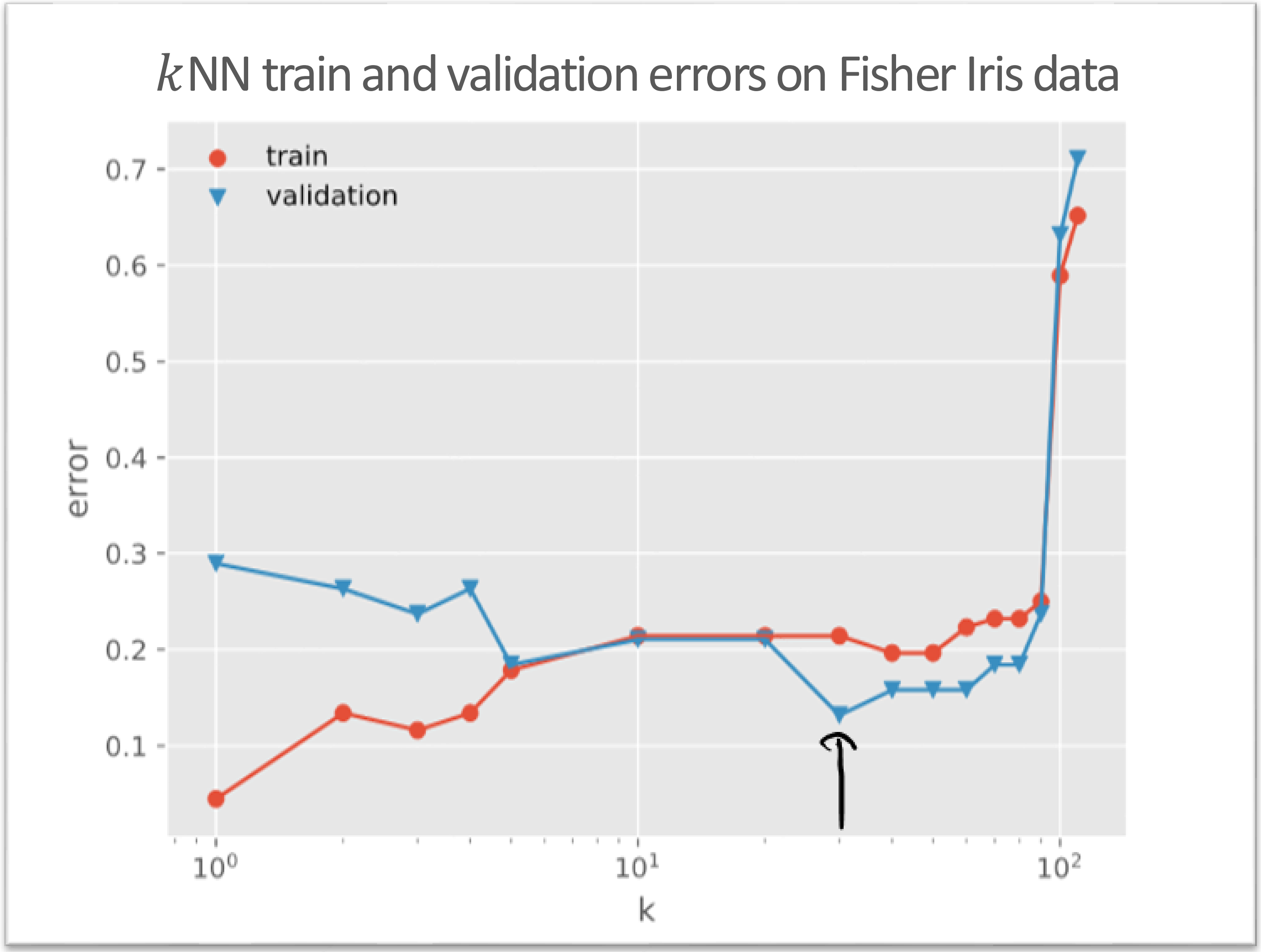


Figure courtesy of Matt Gormley

How should we partition our dataset?

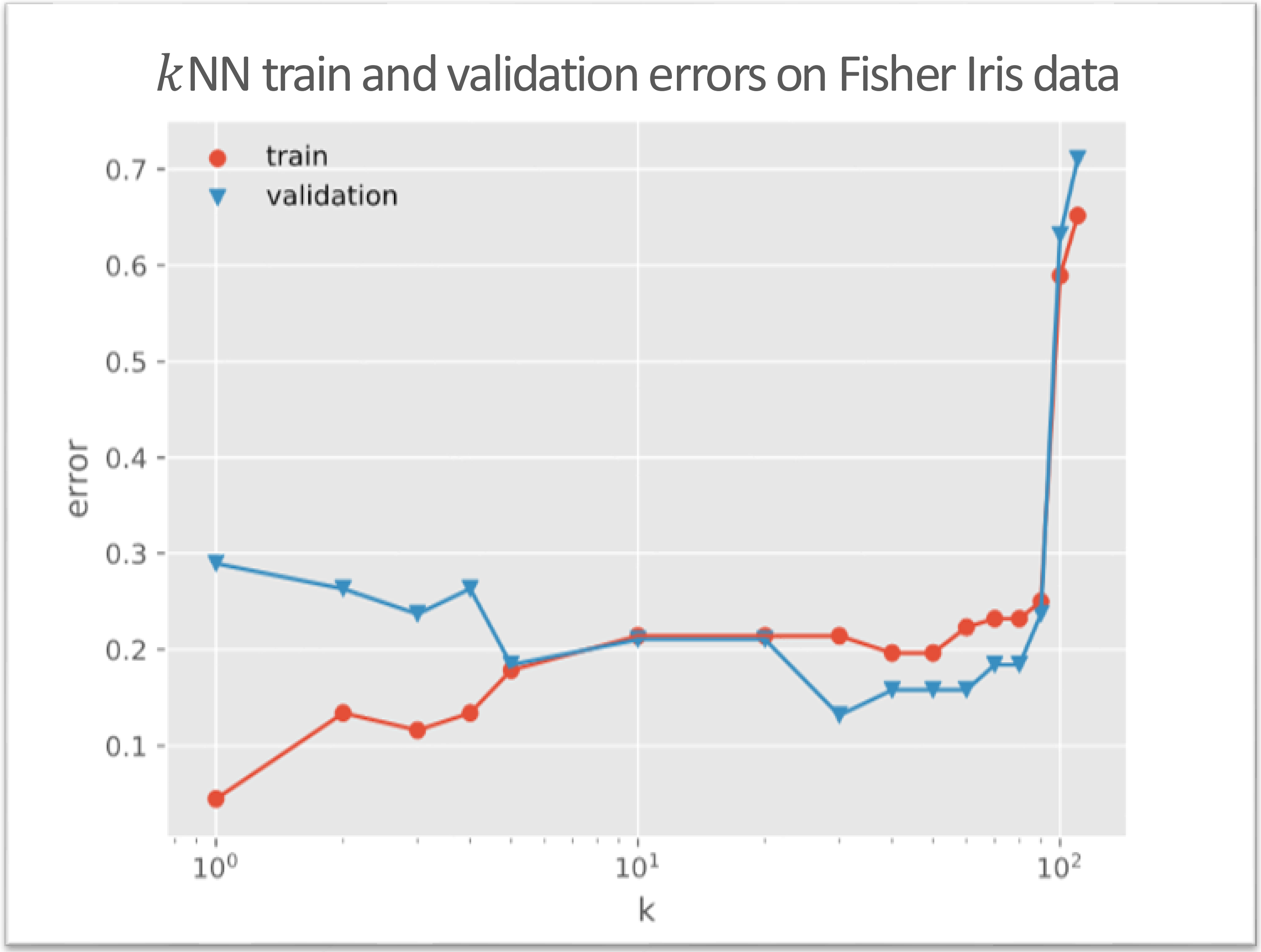
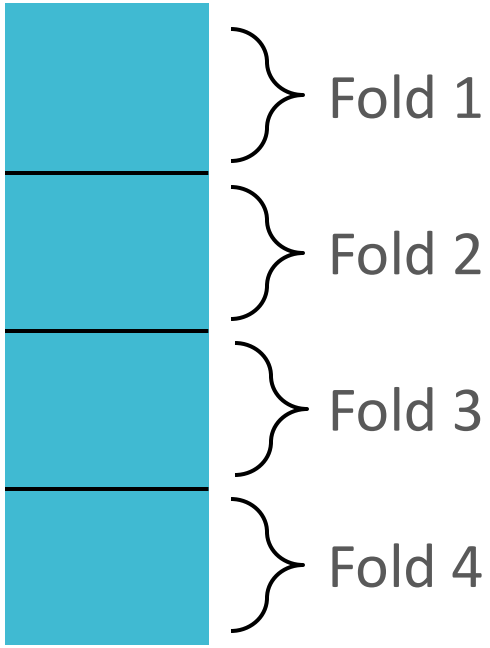


Figure courtesy of Matt Gormley

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:

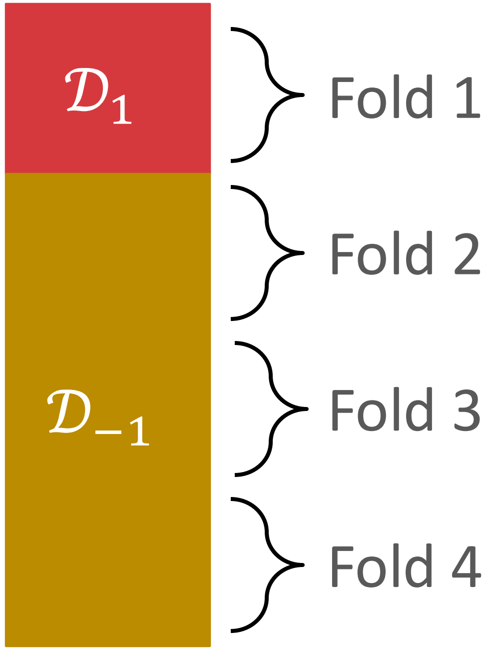


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

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:

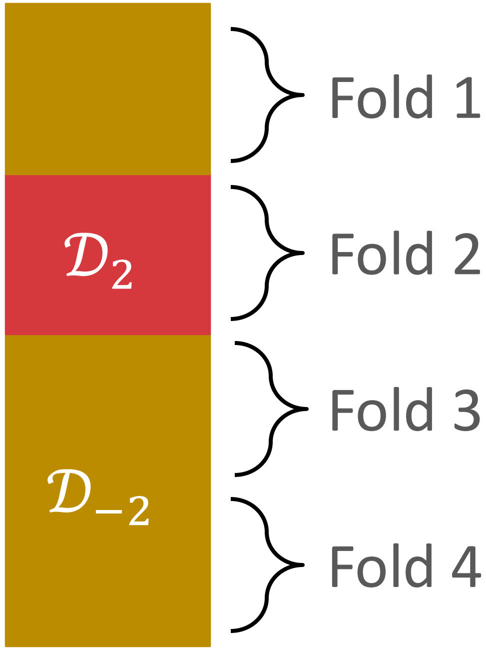


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

K-fold cross-validation

- Given $\mathcal{D}$, split $\mathcal{D}$ into K equally sized datasets or folds: $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K$
- Use each one as a validation set once:

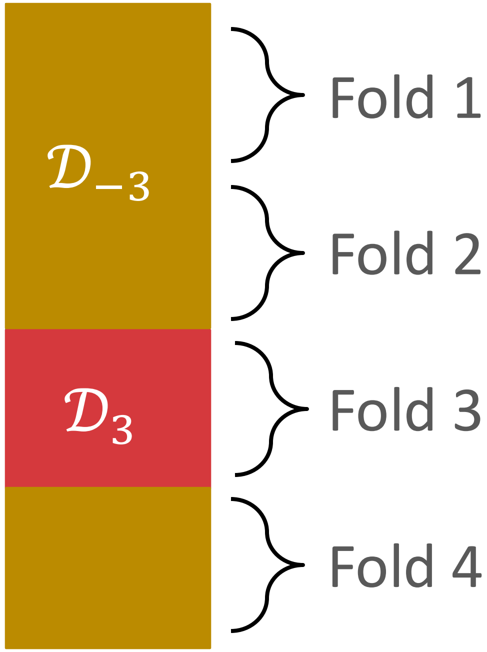


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

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:

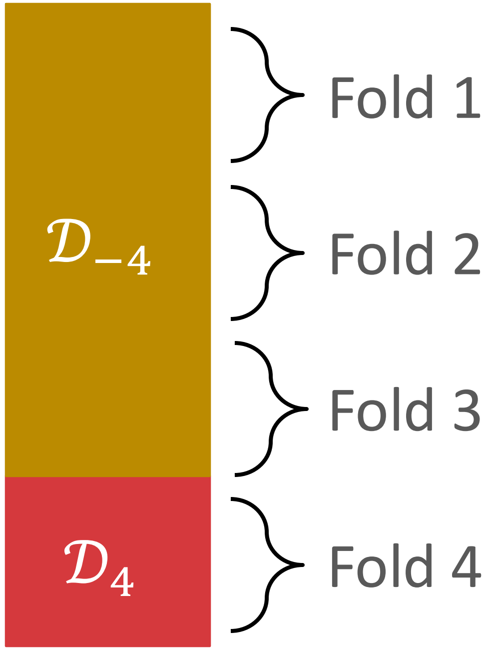


  - Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
  - The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

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:

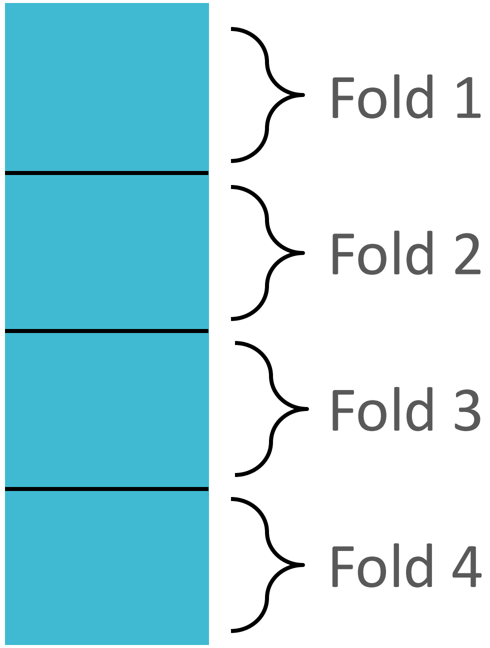


- Let h_{-i} be the classifier learned using $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

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 $\mathcal{D}_{-i} = \mathcal{D} \backslash \mathcal{D}_i$ (all folds other than $\mathcal{D}_i$) and let $e_i = err(h_{-i}, \mathcal{D}_i)$
- The K-fold cross validation error is

$$err_{cv_K} = \frac{1}{K} \sum_{i=1}^{K} e_i$$

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

+1

Summary

	Input	Output
Training	• training dataset • hyperparameters	• best model parameters
Hyperparameter Optimization	• training dataset • validation dataset	• best hyperparameters
Cross-Validation	• training dataset • validation dataset	• cross-validation error
Testing	• test dataset • classifier	• test error

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:

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

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:

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

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

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:

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

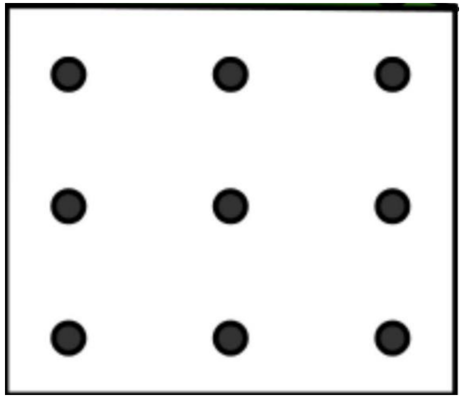
- Train a new model on $\mathcal{D}_{train} \cup \mathcal{D}_{val}$ using $\theta_{\hat{m}}$, $h_{\hat{m}}^{+}$
- Now $err(h_{\hat{m}}^{+}, \mathcal{D}_{test})$ is a good estimate of $err(h_{\hat{m}}^{+})$!

General Methods for Hyperparameter Optimization

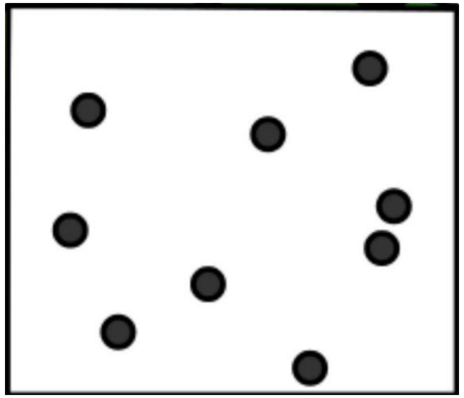
- Idea: set the hyperparameters to optimize some performance metric of the model
- Issue: if we have many hyperparameters that can all take on lots of different values, we might not be able to test all possible combinations
- Commonly used methods:
  - Grid search
  - Random search
  - Bayesian optimization (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

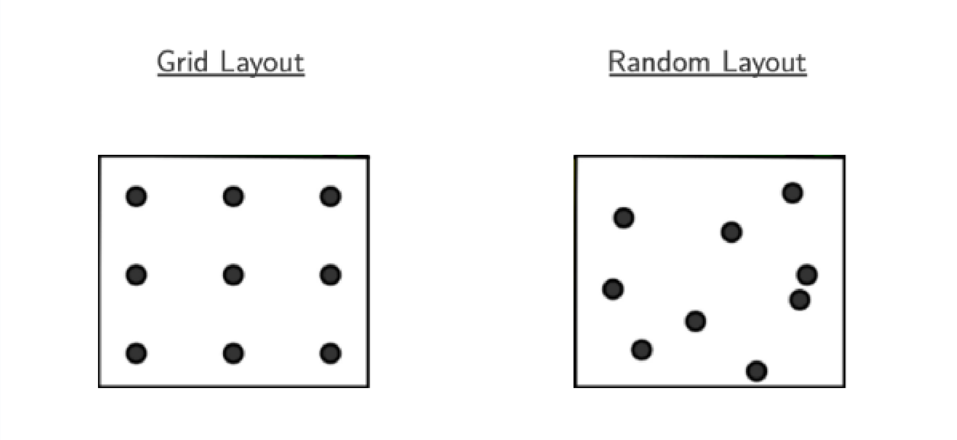


Random Layout



Source: https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

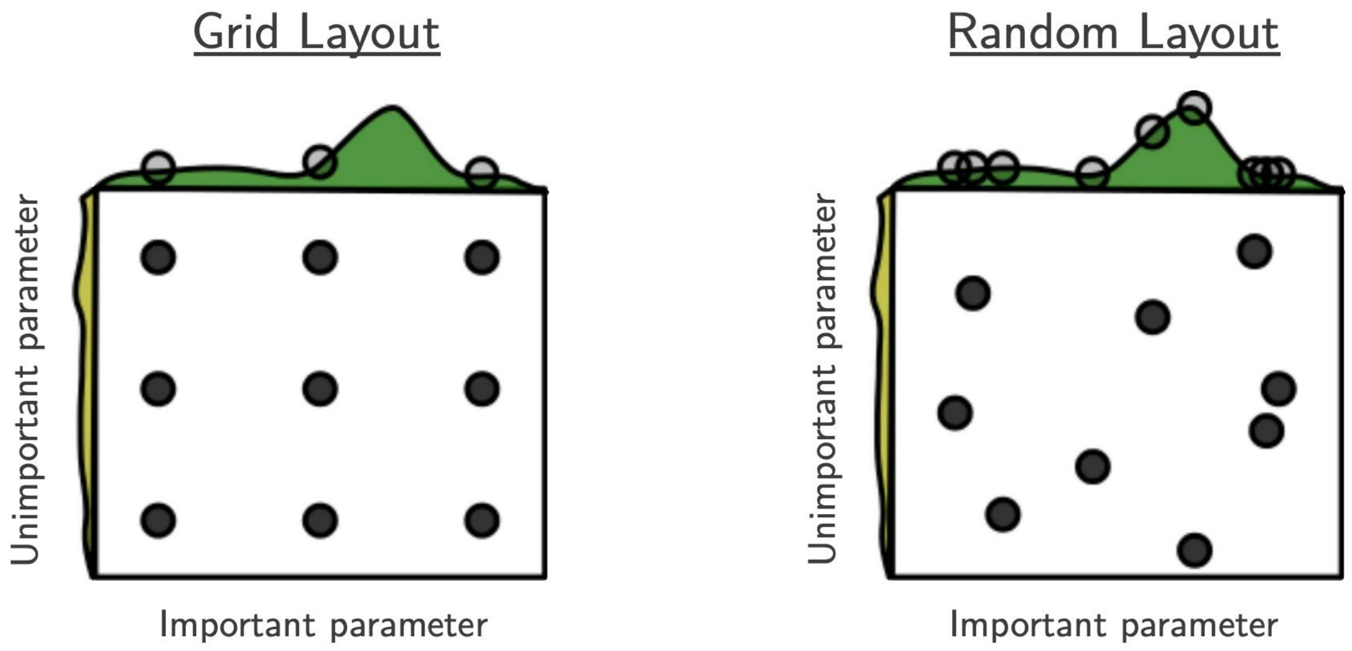
In general, which hyperparameter optimization method do you think will perform better?



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.

Source: https://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

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