Model Selection

Key concepts:

- Hold-out method
- Cross-validation
- Forward and backward feature selection

General Model Selection Problem

- Assume that we have a set of models M=$\{M_1, M_2, \cdots, M_d\}$ that we are trying to select from. Some examples include:
  - **Feature Selection:** each M_i corresponds to using a different feature subset from a large set of potential features
  - **Algorithm Selection:** each M_i corresponds to an algorithm, e.g., Naïve Bayes, Logistic Regression, DT …
  - **Hyperparameter selection/tuning:** each M_i corresponds to a particular parameter choice, e.g., order of polynomial regression, the regularization parameter

Approaches for and related to model selection

- **Empirical methods:** Experimentally determine which model/hyperparameter works best on the specific data in hand
- Theory driven approaches: placing penalty on model complexity, for example
  - Minimum Description Length – "any regularity in a given set of data can be used to compress the data, i.e. to describe it using fewer symbols than needed to describe the data literally. "(Grünwald, 1998)
  - Two-part description:
    - description of the model (complexity) and description of deviation from the model (fit of the data)
- Some times you can avoid model selection and use ensembles
  - Instead of choosing, consider many possibilities and let them vote, or learn to combine their decisions

Can we do model selection using

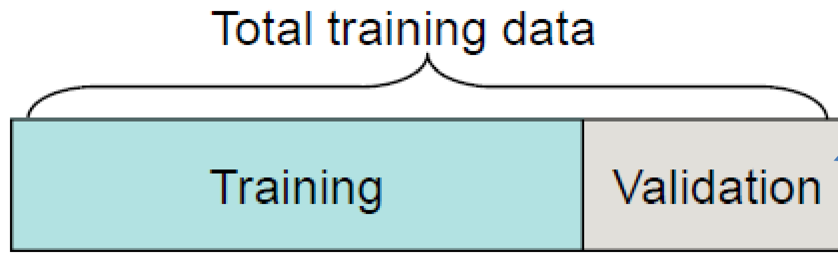
- Training data?
  - Pick the model that gives the best training performance
  - Will pick the most complex model since they can overfit the data
- Test data?
  - Pick the model that gives the best test performance
  - This is cheating --- test data should not be touched during the model building process (which includes model selection)

Appropriate approaches:

- Using a hold-out set (referred to as validation set or a development set) or
- Cross-validation

Simple Holdout Method

Sometimes referred to as **dev**elopment set



1. Divide training set S into S_{train} and S_{valid}
2. Train each model M_i on S_{train} to get a hypothesis h_i
3. Choose and output h_{i^*} with the best performance on S_{valid}

Could retrain the selected model M_{i^*} on S_{train}+ S_{valid} to get the final hypothesis h – this can improve over the original h_{i^*} due to more training data

Advantages/issues of Hold-out method

- Advantage
  - computationally efficient
  - An effective way to choose among nested hypothesis (stopping condition):
    - Deciding when to stop training neural network
    - Deciding when to stop growing a decision tree

- Issues
  - The model selection choice is still made using only the validation data
  - Can be sensitive to the specifics of the validation data
  - Still possible to overfit the validation data since it is a relatively small set of data
  - If we increase validation data size, then we will have less data for training

- To address these problems, we can use **Cross-Validation**

K-fold Cross-validation

- Partition (randomly) training set S into K disjoint subsets $S_1, \cdots, S_K$ (preferably in a class-balanced way)
- To evaluate the cross-validation error of model M_j:

for i=1:K

1. *Train* M_j *on* $S - S_i$ *(*S *removing* S_i*)* $\rightarrow h_{ji}$
2. *Evaluate* h_{ji} *on* $S_i \rightarrow \epsilon_j(i)$

End for

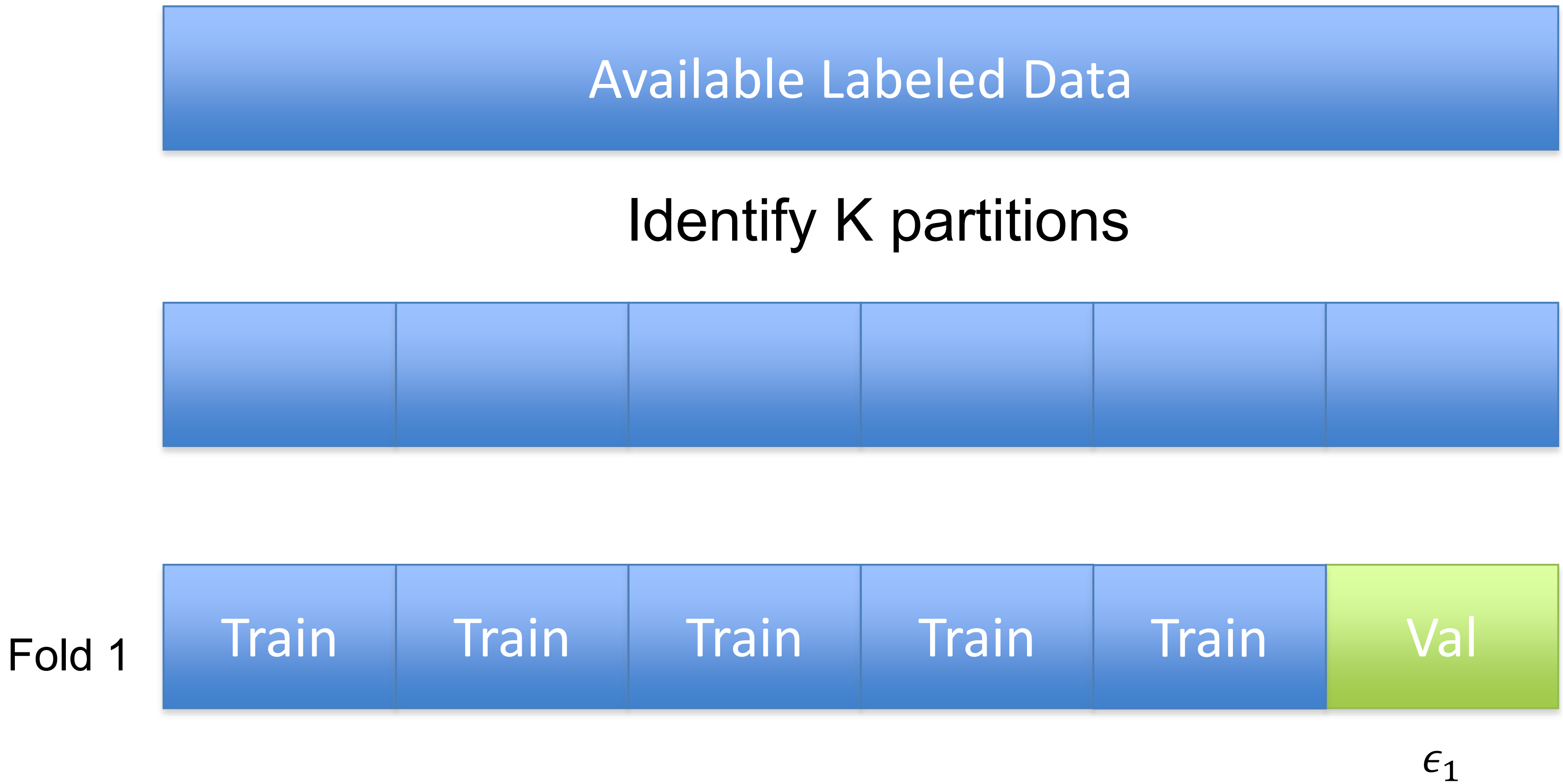
$$\epsilon_j = \frac{1}{K} \sum_i \epsilon_j(i)$$

- Select model that minimizes the cross-validation error:

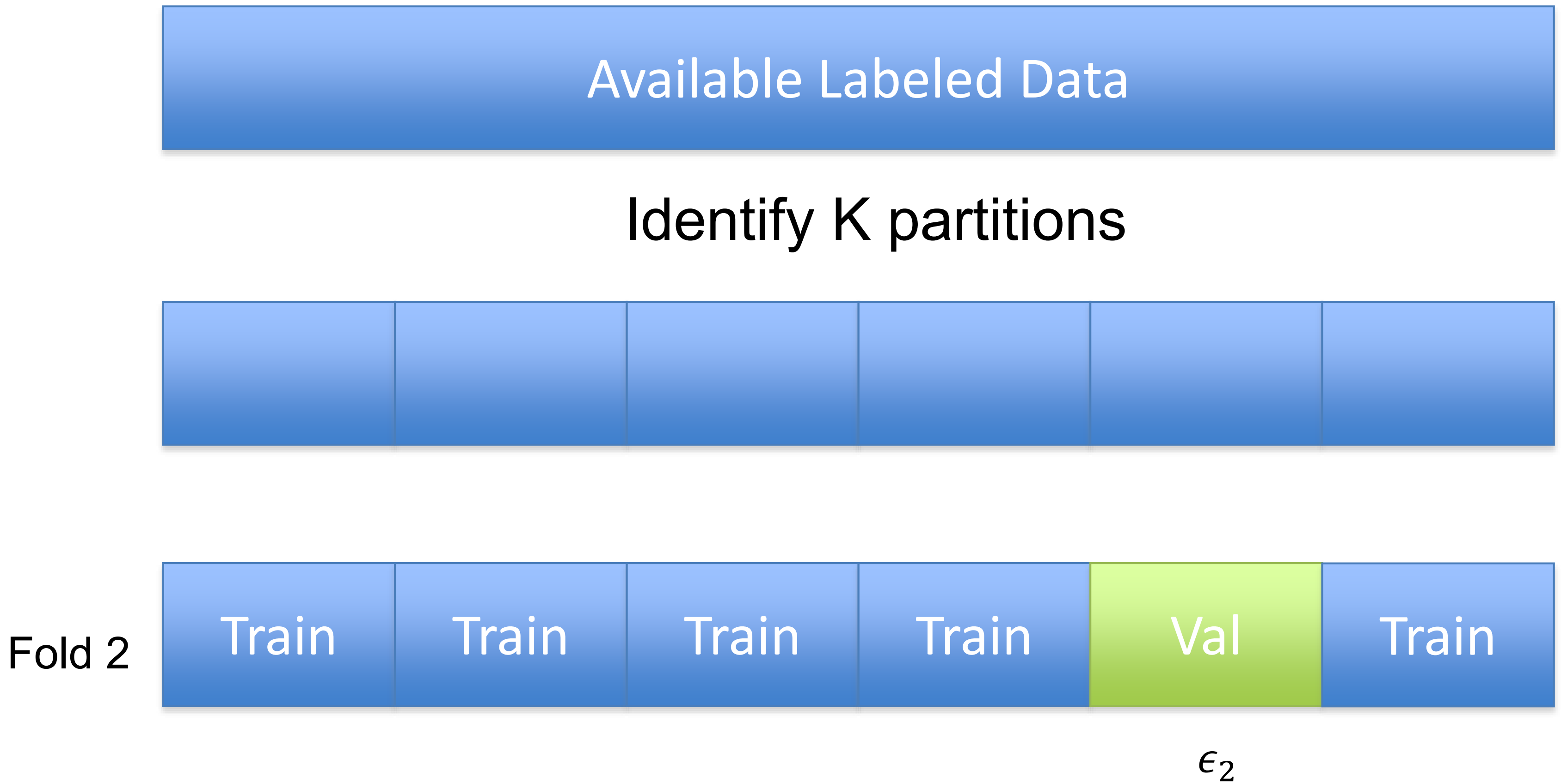
$$M^* = \underset{M_j}{\operatorname{argmin}}\, \epsilon_j$$

- Train M^* on the full training set S and output resulting hypothesis

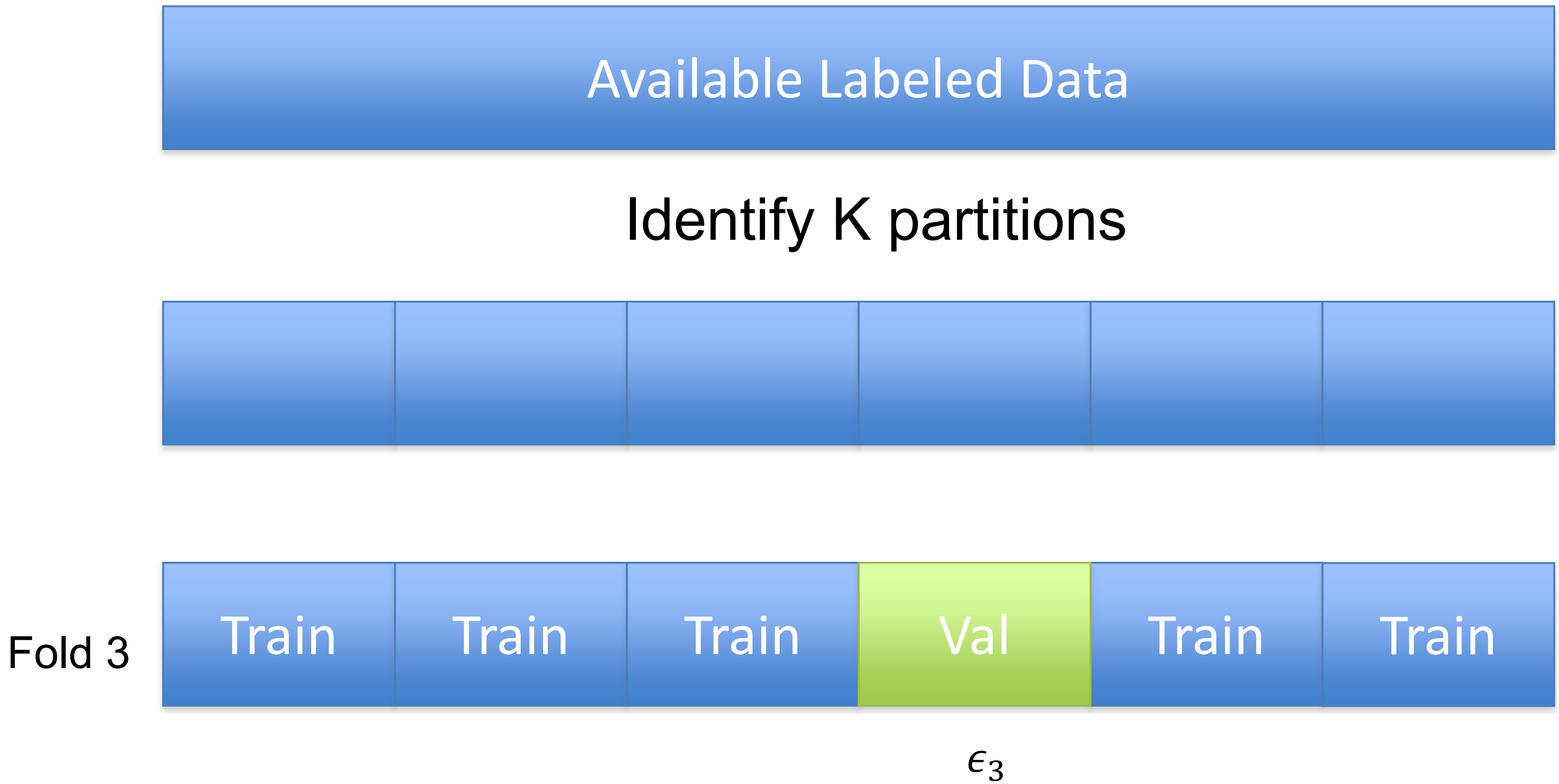
Cross-validation visualized



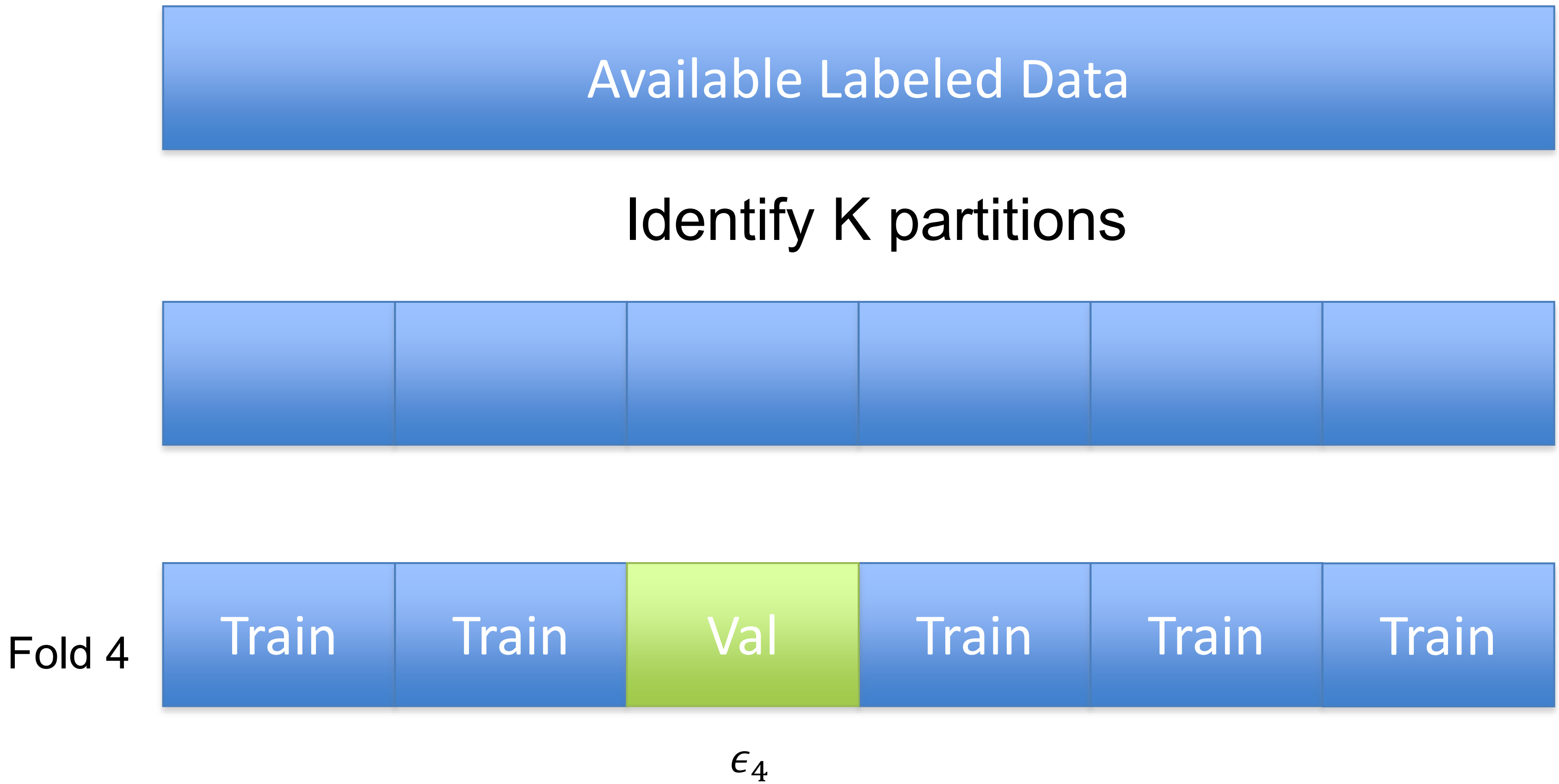
Cross-validation visualized



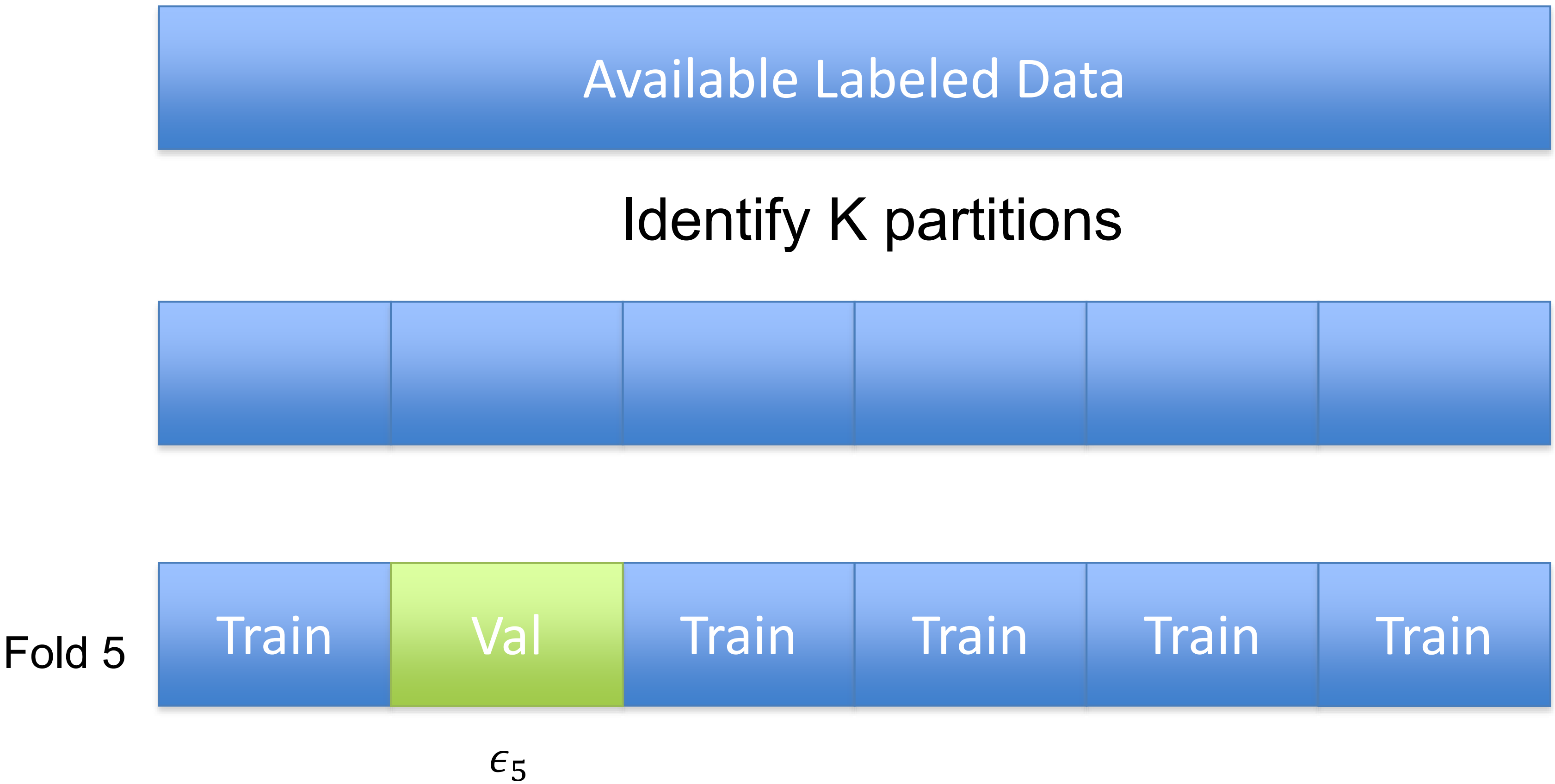
Cross-validation visualized



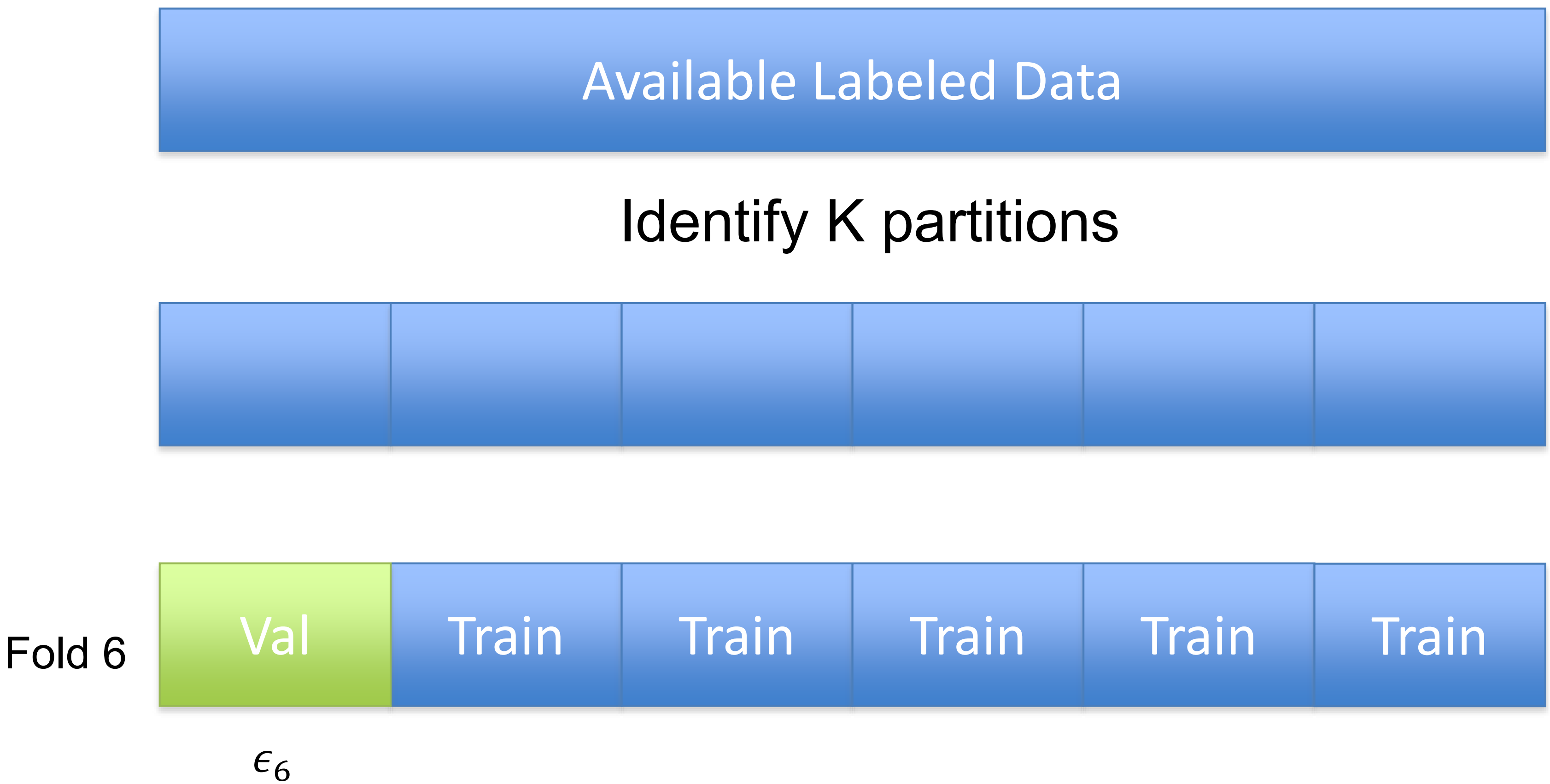
Cross-validation visualized



Cross-validation visualized



Cross-validation visualized



Comments on K-fold Cross-Validation

- Computationally more expensive than simple hold-out method but better use of data
  - Every data point in the training set is used in validating the model selection choices
- If the data is really scarce, we can use the extreme choice of $K = |S|$
  - Each validation set contains only one data point
  - Referred to as Leave-one-out (LOO) cross-validation

Feature Selection

- A special case of model selection problem
- For a given classification problem and the given set of features, it may not be the best to use all the features
  - Features can be noisy and/or irrelevant
  - Features may be redundant

- Feature selection aims to select a subset of features to use. It can
  - Reduce the training/testing time
  - Potentially improve model performance and interpretability

Feature Selection

- Aim to select a subset of features for building the classification model
- Filtering approach:
  - use certain heuristics (e.g., correlation or mutual information with target y) to filter out "irrelevant" features based on the statistics of the data
  - Pro: computationally efficient
  - Con: choice is independent of the choice of the classifier, heuristics are often unreliable
- Wrapper approach
  - Wrap the selection process around a particular classifier (e.g., logistic regression)
  - Aim to select the "optimal" feature subset for that classifier
  - Use hold out or cross-validation to evaluate the feature subsets with the designated classifier
  - Pro: pick the best subset based on empirical performance for particular classifier
  - Con: computationally expensive

Feature selection via search

- A brute-force approach:
  - Evaluate each possible subset with the specific classifier using holdout or cross-validation
  - Select the best subset
- Issue:
  - n features : 2^n possible subsets
  - too big to exhaustively evaluate

- Practically, greedy search-based methods are often used

Forward search for feature selection

- Initialize $F = \phi$ // F represents the selected set, start empty
- Repeat{

  for each feature $i \notin F$ //go over all features not included in F

  let $F_i = F \cup \{i\}$

  evaluate feature set F_i on holdout or cross-validation

  $F =$ best of F_i //select the best feature to add to F

  } until $|F| = k$ //k is the target size

This is a limited-depth best-first search
One can also do a full-depth search and select the best feature subset encountered during search.

Question: will this always choose all features?

Backward search for feature selection

- Initialize F =all features // F represents the selected set, start full
- Repeat{

    for each feature $i \in F$ //go over all features included in F for elimination

        let $F_i = F \setminus \{i\}$

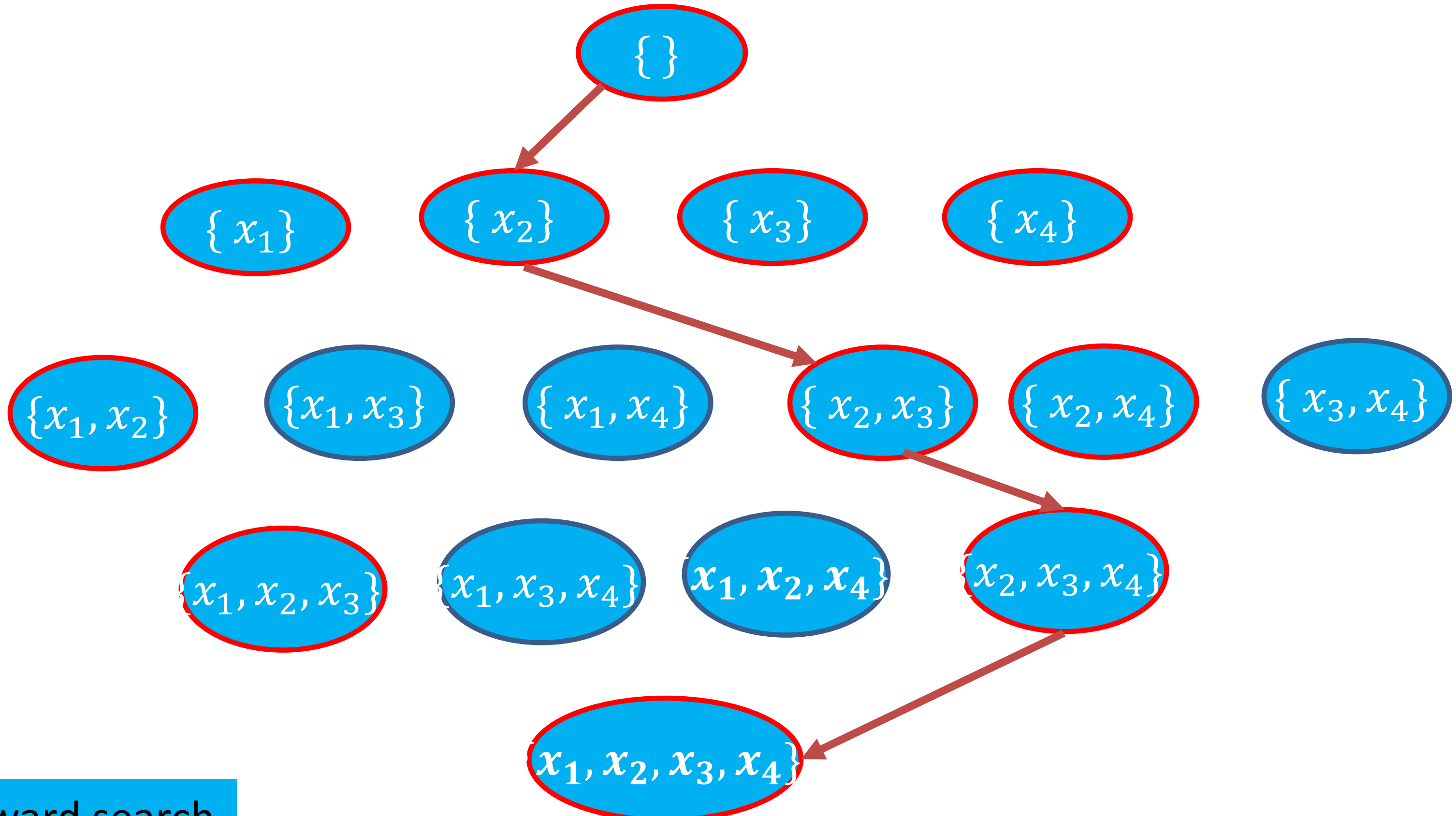
        evaluate feature set F_i using holdout or cross-validation

    F = best of F_i //select the best feature to remove from F

  } until $|F| = k$ //k is the target size

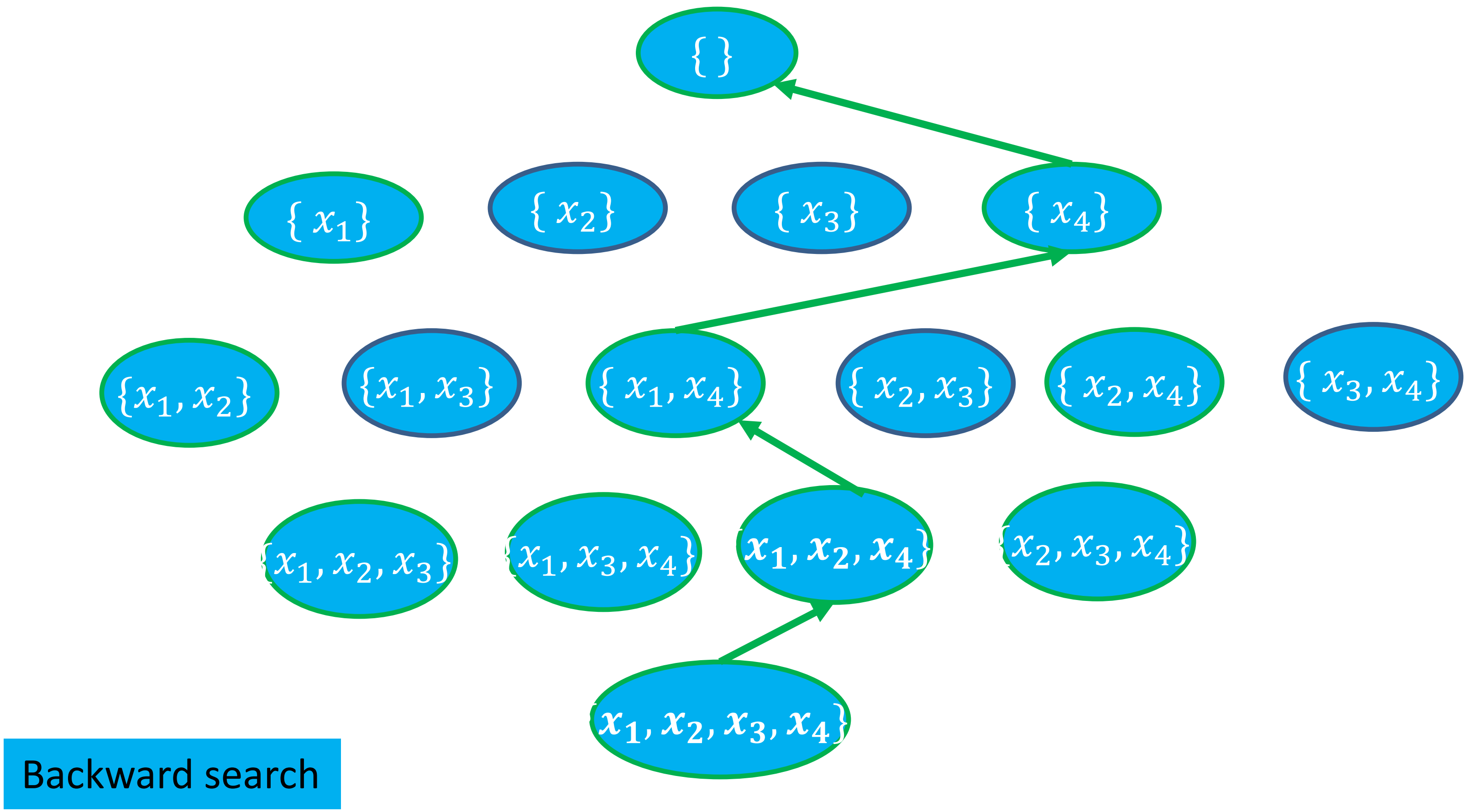
Similar to Forward search, this is a limited depth best-first search
One can perform full-depth best first search and select the best feature subset encountered during search

- Forward and Backward search visit different part of the search space



Forward search

- Forward and Backward search visit different part of the search space



Question

Consider a classification problem with five features, but the label y is only dependent on the sum of the first two features x_1 and x_2. Further, if using x_1 or x_2 separately, we can predict y well. Which search method would more likely give us better result?

A. Forward search

B. Backward search

Summary

- Empirical methods for model selection
  - Simple hold-out:
    - cheap method,
    - useful when there are abundant training data and
    - for deciding stopping conditions
  - K-fold Cross-validation:
    - computationally intensive,
    - useful for limited training data.
    - Large K -> more expensive but better estimate of model performance
- Wrapper Feature Selection
  - Forward and backward search perform greedy best-first search but start with different initial state and different search operator (adding vs. removing)
  - Forward search
    - Work more with smaller feature set during search, more efficient
    - Can miss important features when they need to work in combination
  - Backward search
    - Start with full feature set, can be expensive if it is large
    - Better at capturing combinatorial effect