Hyperparameters and Model Selection
Basic supervised learning framework assumes that there is a single ML algorithm that we will use.

In practice, we may want to consider many different ML algorithms.
Many ML algorithms have hyperparameters:
parameters of ML algorithm \(\neq\) parameters of predictor learned by ML algorithm

E.g., choice of stopping criterion in greedy heuristic for decision trees

Variations in feature representations & predictor templates also regarded as hyperparameters
E.g., allowed forms of predicates in decision trees

Question:
Is there a data-driven way to set hyperparameters?

Answer:
"Model selection" (a.k.a. hyperparameter tuning)

Pretend ML algorithm with different hyperparameter settings are different ML algorithms

\(A_1, A_2, A_3, \ldots\)

Given the same input data, they would learn some (possibly different) predictors \(f_1, f_2, f_3, \ldots\)

How do we decide which ML algorithm to use?
Many ML algorithms have **hyperparameters:**
parameters of ML algorithm (≠ parameters of predictor learned by ML algorithm)

- E.g., choice of stopping criterion in greedy heuristic for decision trees
Hyperparameters and “model selection”

- Many ML algorithms have **hyperparameters**: parameters of ML algorithm \( \neq \) parameters of predictor learned by ML algorithm
  - E.g., choice of stopping criterion in greedy heuristic for decision trees
  - Variations in feature representations & predictor templates also regarded as hyperparameters

Question: Is there a data-driven way to set hyperparameters?

Answer: “Model selection” (a.k.a. hyperparameter tuning)

- Pretend ML algorithm with different hyperparameter settings are different ML algorithms
- Given the same input data, they would learn some (possibly different) predictors

How do we decide which ML algorithm to use?
Hyperparameters and “model selection”

- Many ML algorithms have **hyperparameters**: parameters of ML algorithm ($\neq$ parameters of predictor learned by ML algorithm)
  - E.g., choice of stopping criterion in greedy heuristic for decision trees
- Variations in feature representations & predictor templates also regarded as hyperparameters
  - E.g., allowed forms of predicates in decision trees
Many ML algorithms have **hyperparameters**: parameters of ML algorithm ($\neq$ parameters of predictor learned by ML algorithm)

- E.g., choice of stopping criterion in greedy heuristic for decision trees
- Variations in feature representations & predictor templates also regarded as hyperparameters
  - E.g., allowed forms of predicates in decision trees

**Question**: Is there a data-driven way to set hyperparameters?
Many ML algorithms have **hyperparameters**: parameters of ML algorithm (≠ parameters of predictor learned by ML algorithm)

- E.g., choice of stopping criterion in greedy heuristic for decision trees
- Variations in feature representations & predictor templates also regarded as hyperparameters
  - E.g., allowed forms of predicates in decision trees

**Question:** Is there a data-driven way to set hyperparameters?

**Answer:** “**Model selection**” (a.k.a. **hyperparameter tuning**)

- Pretend ML algorithm with different hyperparameter settings are **different ML algorithms** $A_1, A_2, A_3, \ldots$

- Given the same input data, they would learn some (possibly different) predictors $f_1, f_2, f_3, \ldots$

- How do we decide which ML algorithm to use?
Problematic approach to model selection

A problematic approach:
1. Run ML algorithms $A_1, A_2, A_3, \ldots$ on training data to get predictors $f_1, f_2, f_3, \ldots$.
2. Evaluate $f_1, f_2, f_3, \ldots$ on test data, and return the best predictor (e.g., lowest test error rate).

Flaw: The "test data" are being used for training.

Evaluation of learned predictor is carried out on test data.
A problematic approach:

1. Run ML algorithms $A_1, A_2, A_3, \ldots$ on training data to get predictors $f_1, f_2, f_3, \ldots$

2. Evaluate $f_1, f_2, f_3, \ldots$ on test data, and return the best predictor (e.g., lowest test error rate)

Evaluation of learned predictor is carried out on test data
Problematic approach to model selection

A problematic approach:

1. Run ML algorithms $A_1, A_2, A_3, \ldots$ on training data to get predictors $f_1, f_2, f_3, \ldots$
2. Evaluate $f_1, f_2, f_3, \ldots$ on test data, and return the best predictor (e.g., lowest test error rate)

**Flaw:** The “test data” are being used for training
The end
Cross Validation
Model selection by cross validation

Cross validation: model selection method that simulates training/testing using only training data
Model selection by cross validation

**Cross validation**: model selection method that simulates training/testing using only training data

1. Given training data $S$, split into two parts, $S_1$ and $S_2$ in some manner (e.g., randomly)
Model selection by cross validation

Cross validation: model selection method that simulates training/testing using only training data

1. Given training data $S$, split into two parts, $S_1$ and $S_2$ in some manner (e.g., randomly)
2. Run ML algorithms $A_1, A_2, A_3, \ldots$ on $S_1$ to get predictors $f_1, f_2, f_3, \ldots$
Model selection by cross validation

**Cross validation:** model selection method that simulates training/testing using only training data

1. Given training data $S$, split into two parts, $S_1$ and $S_2$ in some manner (e.g., randomly)
2. Run ML algorithms $A_1, A_2, A_3, \ldots$ on $S_1$ to get predictors $f_1, f_2, f_3, \ldots$
3. Evaluate $f_1, f_2, f_3, \ldots$ on $S_2$ (e.g., in terms of error rate on $S_2$)
   - Let $i^* \in \{1, 2, \ldots\}$ be the index of the best predictor as above
Model selection by cross validation

**Cross validation:** model selection method that simulates training/testing using only training data

1. Given training data $S$, split into two parts, $S_1$ and $S_2$ in some manner (e.g., randomly)
2. Run ML algorithms $A_1, A_2, A_3, \ldots$ on $S_1$ to get predictors $f_1, f_2, f_3, \ldots$
3. Evaluate $f_1, f_2, f_3, \ldots$ on $S_2$ (e.g., in terms of error rate on $S_2$)
   ▶ Let $i^* \in \{1, 2, \ldots\}$ be the index of the best predictor as above
4. Return $A_{i^*}$ as the selected ML algorithm

![Diagram of cross validation process](image)
Model selection by cross validation

**Cross validation**: model selection method that simulates training/testing using only training data

1. Given training data \( S \), split into two parts, \( S_1 \) and \( S_2 \) in some manner (e.g., randomly)
2. Run ML algorithms \( A_1, A_2, A_3, \ldots \) on \( S_1 \) to get predictors \( f_1, f_2, f_3, \ldots \)
3. Evaluate \( f_1, f_2, f_3, \ldots \) on \( S_2 \) (e.g., in terms of error rate on \( S_2 \))
   - Let \( i^* \in \{1, 2, \ldots \} \) be the index of the best predictor as above
4. Return \( A_{i^*} \) as the selected ML algorithm

(Final step: Run \( A_{i^*} \) on the training data to get final predictor \( f \), which can be evaluated on test data)
Variants of cross validation: \( K \)-fold cross validation

\( K \)-fold cross validation:

- Each \( A_i \) is run and evaluated \( K \) times
- Select the \( A_i \) with best average evaluation
Variants of cross validation: Leave-one-out cross validation

**Leave-one-out cross validation:**

▶ $K$-fold cross validation with $K = n$
▶ Pro: $A_i$ is always run on $n - 1$ of the training data
▶ Con: Running each algorithm $n$ times can be expensive
Consider the training/testing setup that makes sense for the application
Domain-specific cross validation

Consider the training/testing setup that makes sense for the application

**Example:** Time-ordered data

- Suppose there is reason to believe that examples from the “past” are not interchangeable with examples from the “future”
Consider the training/testing setup that makes sense for the application

**Example:** Time-ordered data

- Suppose there is reason to believe that examples from the “past” are not interchangeable with examples from the “future”
  - I.e., only makes sense to train on examples from “past”, and test on examples from “future”
Domain-specific cross validation

Consider the training/testing setup that makes sense for the application

**Example:** Time-ordered data

- Suppose there is reason to believe that examples from the “past” are not interchangeable with examples from the “future”
  - I.e., only makes sense to train on examples from “past”, and test on examples from “future”

Which form of cross validation makes sense?
For model selection, cross validation simulates training/testing **using only training data**
For model selection, cross validation simulates training/testing using only training data

- Useful whenever ML algorithm has hyperparameters (pretty much all of them)
Discussion

For model selection, cross validation simulates training/testing using only training data

- Useful whenever ML algorithm has hyperparameters (pretty much all of them)
- **Ideally**: Test data is never looked at until development is done and it is time for evaluation
Discussion

For model selection, cross validation simulates training/testing using only training data

- Useful whenever ML algorithm has hyperparameters (pretty much all of them)
- **Ideally:** Test data is never looked at until development is done and it is time for evaluation
- **In practice:** Development process is iterative, and there is inevitably “leakage” of information from test data into development choices
For model selection, cross validation simulates training/testing **using only training data**

- Useful whenever ML algorithm has hyperparameters (pretty much all of them)
- **Ideally:** Test data is never looked at until development is done and it is time for evaluation
- **In practice:** Development process is iterative, and there is inevitably “leakage” of information from test data into development choices
  - **Where possible:** Periodically acquire new test data so that reliable evaluations are possible
The end