Stages of (Batch) Machine Learning

**Given:** labeled training data $X, Y = \left\{ \langle x^{(i)}, y^{(i)} \rangle \right\}_{i=1}^{n}$

- Assumes each $x^{(i)} \sim D(X)$ with $y^{(i)} = f_{target}(x^{(i)})$

**Train the model:**

$\text{model} \leftarrow \text{classifier.train}(X, Y)$

**Apply the model to new data:** $x \sim D(X)$

- Given: new unlabeled instance

$y_{\text{prediction}} \leftarrow \text{model.predict}(x)$
Classification Metrics

\[
\text{accuracy} = \frac{\# \text{ correct predictions}}{\# \text{ test instances}}
\]

\[
\text{error} = 1 - \text{accuracy} = \frac{\# \text{ incorrect predictions}}{\# \text{ test instances}}
\]
## Confusion Matrix

- Given a dataset of $P$ positive instances and $N$ negative instances:
  
<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Predicted Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes, No</td>
</tr>
<tr>
<td></td>
<td>TP, FN</td>
</tr>
<tr>
<td>No</td>
<td>FP, TN</td>
</tr>
</tbody>
</table>

  \[
  \text{accuracy} = \frac{TP + TN}{P + N}
  \]

- Imagine using classifier to identify positive cases (i.e., for information retrieval)

  \[
  \text{precision} = \frac{TP}{TP + FP} \quad \text{recall} = \frac{TP}{TP + FN}
  \]

  Probability that a randomly selected result is relevant
  Probability that a randomly selected relevant document is retrieved
Training Data and Test Data

• Training data: data used to build the model
• Test data: new data, not used in the training process

• Training performance is often a poor indicator of generalization performance
  – Generalization is what we really care about in ML
  – Easy to overfit the training data
  – Performance on test data is a good indicator of generalization performance
  – i.e., test accuracy is more important than training accuracy
Simple Decision Boundary

TWO-CLASS DATA IN A TWO-DIMENSIONAL FEATURE SPACE

Decision Region 1
Decision Region 2
Decision Boundary
More Complex Decision Boundary

TWO-CLASS DATA IN A TWO-DIMENSIONAL FEATURE SPACE

Decision Region 1

Decision Region 2

Decision Boundary
Example: The Overfitting Phenomenon
A Complex Model

Y = high-order polynomial in X
The True (simpler) Model

\[ Y = a \, X + b + \text{noise} \]
How Overfitting Affects Prediction

Predictive Error

Error on Training Data

Model Complexity

Slide by Padhraic Smyth, UCIrvine
How Overfitting Affects Prediction

Predictive Error versus Model Complexity

- Error on Training Data
- Error on Test Data

Slide by Padhraic Smyth, UCIrvine
How Overfitting Affects Prediction

Predictive Error

Underfitting

Overfitting

Error on Test Data

Error on Training Data

Model Complexity

Ideal Range for Model Complexity

Predictive Error

Overfitting

Error on Test Data

Error on Training Data

Model Complexity

Ideal Range for Model Complexity

Predictive Error

Overfitting

Error on Test Data

Error on Training Data

Model Complexity

Ideal Range for Model Complexity

Predictive Error

Overfitting

Error on Test Data

Error on Training Data

Model Complexity

Ideal Range for Model Complexity

Predictive Error

Overfitting

Error on Test Data

Error on Training Data

Model Complexity

Ideal Range for Model Complexity
Comparing Classifiers

Say we have two classifiers, $C1$ and $C2$, and want to choose the best one to use for future predictions.

Can we use training accuracy to choose between them?

• No!
  – e.g., $C1 =$ pruned decision tree, $C2 =$ 1-NN

    training_accuracy(1-NN) = 100%, but may not be best

Instead, choose based on test accuracy...
Training and Test Data

Idea:
Train each model on the “training data”...

...and then test each model’s accuracy on the test data
\textit{k-}Fold Cross-Validation

• Why just choose one particular \textit{“split”} of the data?
  – In principle, we should do this multiple times since performance may be different for each split

• \textit{k-}Fold Cross-Validation (e.g., \(k=10\))
  – randomly partition full data set of \(n\) instances into \(k\) disjoint subsets (each roughly of size \(n/k\))
  – Choose each fold in turn as the test set; train model on the other folds and evaluate
  – Compute statistics over \(k\) test performances, or choose best of the \(k\) models
  – Can also do \textit{“leave-one-out CV”} where \(k = n\)
Example 3-Fold CV

- Full Data Set
- 1st Partition
  - Training Data
  - Test Data
- 2nd Partition
  - Training Data
  - Test Data
- kth Partition
  - Training Data
  - Test Data

Test Performance
Test Performance
Test Performance

Summary statistics over k test performances
Optimizing Model Parameters

Can also use CV to choose value of model parameter $P$

- Search over space of parameter values $p \in \text{values}(P)$
  - Evaluate model with $P = p$ on validation set
- Choose value $p'$ with highest validation performance
- Learn model on full training set with $P = p'$

Choose value of $p$ of the model with the best validation performance
More on Cross-Validation

• Cross-validation generates an approximate estimate of how well the classifier will do on “unseen” data
  – As $k \rightarrow n$, the model becomes more accurate (more training data)
  – ...but, CV becomes more computationally expensive
  – Choosing $k < n$ is a compromise

• Averaging over different partitions is more robust than just a single train/validate partition of the data

• It is an even better idea to do CV repeatedly!
Multiple Trials of $k$-Fold CV

1.) Loop for $t$ trials:

   a.) Randomize Data Set

   b.) Perform $k$-fold CV

2.) Compute statistics over $t \times k$ test performances
Comparing Multiple Classifiers

1.) Loop for $t$ trials:

   a.) Randomize Data Set

   b.) Perform $k$-fold CV

2.) Compute statistics over $t \times k$ test performances

   Test each candidate learner on same training/testing splits

   Allows us to do paired summary statistics (e.g., paired t-test)
Learning Curve

- Shows performance versus the # training examples
  - Compute over a single training/testing split
  - Then, average across multiple trials of CV
Building Learning Curves

1.) Loop for $t$ trials:
   a.) Randomize Data Set
      - Shuffle Full Data Set
   b.) Perform $k$-fold CV

2.) Compute learning curve over each training/testing split
   - For each partition:
     - Compute learning curve over each training/testing split
   - Loop for $t$ trials:
     - Curve $C_1$
     - Curve $C_2$

2.) Compute statistics over $t \times k$ learning curves