These slides were assembled by Byron Boots, with only minor modifications from Eric Eaton’s slides and grateful acknowledgement to the many others who made their course materials freely available online. Feel free to reuse or adapt these slides for your own academic purposes, provided that you include proper attribution.

Evaluation
Stages of (Batch) Machine Learning

**Given:** labeled training data $X, Y = \{\langle x_i, y_i \rangle \}_{i=1}^{n}$

- Assumes each $x_i \sim \mathcal{D}(\mathcal{X})$ with $y_i = f_{\text{target}}(x_i)$

**Train the model:**

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

**Apply the model to new data:**

- Given: new unlabeled instance $x \sim \mathcal{D}(\mathcal{X})$
  
  $y_{\text{prediction}} \leftarrow \text{model}.\text{predict}(x)$
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>TP</td>
</tr>
<tr>
<td>No</td>
<td>FP</td>
</tr>
</tbody>
</table>

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

- Imagine using classifier to identify positive cases (there is a cat in an image):

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

Probability that classifier predicts positive correctly

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

Probability that actual class is predicted correctly
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

Slide by Padhraic Smyth, UCIrvine
More Complex Decision Boundary

TWO-CLASS DATA IN A TWO-DIMENSIONAL FEATURE SPACE

Feature 1
Feature 2
Decision Region 1
Decision Region 2
Decision Boundary

Slide by Padhraic Smyth, UCIrvine
Example: The Overfitting Phenomenon

![Diagram of a scatter plot with the axes labeled X and Y. The points show a trend in the data.](image)
A Complex Model

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

\[ Y = a \times X + b + \text{noise} \]
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

Error on Test Data

Error on Training Data

Model Complexity

Slide by Padhraic Smyth, UCIrvine
How Overfitting Affects Prediction

Predictive Error

Underfitting

Overfitting

Error on Training Data

Error on Test Data

Ideal Range for Model Complexity

Model Complexity

Slide by Padhraic Smyth, UCIrvine
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
    \[
    \text{training\_accuracy}(1\text{-NN}) = 100\%, \text{ but may not be best}
    \]

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
**k-Fold Cross-Validation**

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

• **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 “leave-one-out CV” where \( k = n \)
Example 3-Fold CV

Full Data Set

1\textsuperscript{st} Partition

\textbf{Test Data}

\textbf{Training Data}

2\textsuperscript{nd} Partition

\textbf{Training Data}

\textbf{Test Data}

\textbf{Training Data}

\textbf{Test Data}

\textbf{...}

k\textsuperscript{th} Partition

\textbf{Training Data}

\textbf{Test Data}

\textbf{Test Performance}

\textbf{Test Performance}

\textbf{Test Performance}

Summary statistics over k test performances
More on Cross-Validation

• Cross-validation generates an approximate estimate of how well the classifier will do on “unseen” data
  – As \( k \to n \), 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
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

a.) Randomize Data Set

b.) Perform k-fold CV