Lecture 7

Machine Learning

Validation

and

Cross-Validation
• Performance evaluation and model selection methods
  • validation
  • cross-validation
    • k-fold
    • Leave-one-out
• In this lecture, it’s convenient to show examples in the context of regression

• In regression, labels $y_i$ are continuous

• Classification/regression are solved very similarly

• Everything we have done so far transfers to regression with very minor changes

• Error: sum of distances from examples to the fitted model
Training/Test Data Split

• Talked about splitting data in training/test sets
  • training data is used to fit parameters
  • test data is used to assess how classifier generalizes to new data

• What if classifier has “non-tunable” parameters?
  • a parameter is “non-tunable” if tuning (or training) it on the training data leads to overfitting

• Examples:
  • k in kNN classifier
  • number of hidden units in MNN
  • number of hidden layers in MNN
  • etc...
• Want to fit a polynomial machine $f(x,w)$
• Instead of fixing polynomial degree, make it parameter $d$
  • learning machine $f(x,w,d)$
• Consider just three choices for $d$
  • degree 1
  • degree 2
  • degree 3
• Training error is a bad measure to choose $d$
  • degree 3 is the best according to the training error, but overfits the data
• What about test error? Seems appropriate
  • degree 2 is the best model according to the test error
• Except what do we report as the test error now?
• Test error should be computed on data that was not used for training at all
• Here used “test” data for training, i.e. choosing model
- Same question when choosing among several classifiers
- Our polynomial degree example can be looked at as choosing among 3 classifiers (degree 1, 2, or 3)
- Solution: split the labeled data into three parts

<table>
<thead>
<tr>
<th>labeled data</th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>≈60%</td>
<td>≈20%</td>
<td>≈20%</td>
</tr>
<tr>
<td>train tunable parameters $w$</td>
<td>train other parameters, or to select classifier</td>
<td>use only to assess final performance</td>
<td></td>
</tr>
</tbody>
</table>
Training/Validation/Test

labeled data

<table>
<thead>
<tr>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>≈60%</td>
<td>≈20%</td>
<td>≈20%</td>
</tr>
</tbody>
</table>

Training error: computed on training examples

Validation error: computed on validation examples

Test error: computed on test examples
Training/Validation/Test Data

- **Training Data**
- **Validation Data**
  - $d = 2$ is chosen
- **Test Data**
  - 1.3 test error computed for $d = 2$
Choosing Parameters: Example

- Need to choose number of hidden units for a MNN
  - The more hidden units, the better can fit training data
  - But at some point we overfit the data
Diagnosing Underfitting/Overfitting

**Underfitting**
- large training error
- large validation error

**Just Right**
- small training error
- small validation error

**Overfitting**
- small training error
- large validation error
Fixing Underfitting/Overfitting

- Fixing Underfitting
  - getting more training examples will not help
  - get more features
  - try more complex classifier
    - if using MNN, try more hidden units

- Fixing Overfitting
  - getting more training examples might help
  - try smaller set of features
  - Try less complex classifier
    - If using MNN, try less hidden units
Train/Test/Validation Method

• Good news
  • Very simple

• Bad news:
  • Wastes data
    • in general, the more data we have, the better are the estimated parameters
    • we estimate parameters on 40% less data, since 20% removed for test and 20% for validation data
  • If we have a small dataset our validation set might just be lucky or unlucky
Small Dataset

- “Unlucky” validation set:

  - Linear Model: Mean Squared Error = 2.4
  - Quadratic Model: Mean Squared Error = 0.9
  - Join the dots Model: Mean Squared Error = 2.2
Cross Validation

- Create multiple splits of training/validation
- Average results over splits

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Training</td>
<td>Validation</td>
<td>≈20%</td>
</tr>
<tr>
<td>2</td>
<td>Training</td>
<td>Validation</td>
<td>Training</td>
</tr>
<tr>
<td>3</td>
<td>Training</td>
<td>Validation</td>
<td>Training</td>
</tr>
<tr>
<td>4</td>
<td>Validation</td>
<td>Training</td>
<td>Test</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>≈20%</td>
</tr>
</tbody>
</table>
LOOCV (Leave-one-out Cross Validation)

For \( k=1 \) to \( R \)

1. Let \((x^k, y^k)\) be the \( k \) example
LOOCV (Leave-one-out Cross Validation)

For $k = 1$ to $n$

1. Let $(x^k, y^k)$ be the $k$th example

2. Temporarily remove $(x^k, y^k)$ from the dataset
LOOCV (Leave-one-out Cross Validation)

For $k=1$ to $n$

1. Let $(x^k,y^k)$ be the $k$th example
2. Temporarily remove $(x^k,y^k)$ from the dataset
3. Train on the remaining $n-1$ examples
LOOCV (Leave-one-out Cross Validation)

For \( k=1 \) to \( n \)

1. Let \((x^k, y^k)\) be the \( k \)th example

2. Temporarily remove \((x^k, y^k)\) from the dataset

3. Train on the remaining \( n-1 \) examples

4. Note your error on \((x^k, y^k)\)
For $k=1$ to $n$

1. Let $(x^k, y^k)$ be the $k$th example

2. Temporarily remove $(x^k, y^k)$ from the dataset

3. Train on the remaining $n-1$ examples

4. Note your error on $(x^k, y^k)$

When you’ve done all points, report the mean error
LOOCV (Leave-one-out Cross Validation)

\[ \text{MSE}_{\text{LOOCV}} = 2.12 \]
LOOCV for Quadratic Regression

\[ \text{MSE}_{\text{LOOCV}} = 0.962 \]
LOOCV for Join The Dots

\[ \text{MSE}_{\text{LOOCV}} = 3.33 \]
<table>
<thead>
<tr>
<th></th>
<th><strong>Downside</strong></th>
<th><strong>Upside</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Validation-set</strong></td>
<td>may give unreliable estimate of future performance</td>
<td>cheap</td>
</tr>
<tr>
<td><strong>Leave-one-out</strong></td>
<td>expensive</td>
<td>doesn’t waste data</td>
</tr>
</tbody>
</table>
K-Fold Cross Validation

Randomly break the dataset into $k$ partitions in this example we’ll have $k=3$ partitions colored Red, Green and Blue.
• Randomly break the dataset into $k$ partitions
• In example have $k=3$ partitions colored red green and blue
• For the blue partition: train on all points not in the blue partition. Find test-set sum of errors on blue points
Randomly break the dataset into $k$ partitions

in example have $k=3$ partitions colored red, green, and blue

For the blue partition: train on all points not in the blue partition. Find sum of errors on blue points

For the green partition: train on all points not in green partition. Find sum of errors on green points
• Randomly break the dataset into k partitions
• in example have k=3 partitions colored red green and blue
• For the blue partition: train on all points not in the blue partition. Find sum of errors on blue points
• For the green partition: train on all points not in green partition. Find sum of errors on green points
• For the red partition: train on all points not in red partition. Find sum of errors on red points
K-Fold Cross Validation

- Randomly break the dataset into $k$ partitions
- in example have $k=3$ partitions colored red, green, and blue
- **For the blue partition:** train on all points not in the blue partition. Find sum of errors on blue points
- **For the green partition:** train on all points not in green partition. Find sum of errors on green points
- **For the red partition:** train on all points not in red partition. Find sum of errors on red points
- Report the mean error

Linear Regression

$\text{MSE}_{3\text{FOLD}} = 2.05$
K-Fold Cross Validation

- Randomly break the dataset into k partitions
- in example have k=3 partitions colored red green and blue
- For the blue partition: train on all points not in the blue partition. Find sum of errors on blue points
- For the green partition: train on all points not in green partition. Find sum of errors on green points
- For the red partition: train on all points not in red partition. Find sum of errors on red points
- Report the mean error

Quadratic Regression

$\text{MSE}_{3\text{FOLD}} = 1.11$
K-Fold Cross Validation

- Randomly break the dataset into $k$ partitions
- in example have $k=3$ partitions colored red green and blue
- For the blue partition: train on all points not in the blue partition. Find sum of errors on blue points
- For the green partition: train on all points not in green partition. Find sum of errors on green points
- For the red partition: train on all points not in red partition. Find sum of errors on red points
- Report the mean error

Joint-the-dots

$\text{MSE}_{3\text{FOLD}} = 2.93$
<table>
<thead>
<tr>
<th></th>
<th>Downside</th>
<th>Upside</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation-set</td>
<td>may give unreliable estimate of future performance</td>
<td>cheap</td>
</tr>
<tr>
<td>Leave-one-out</td>
<td>expensive</td>
<td>doesn’t waste data</td>
</tr>
<tr>
<td>10-fold</td>
<td>wastes 10% of the data, 10 times more expensive than validation set</td>
<td>only wastes 10%, only 10 times more expensive instead of n times</td>
</tr>
<tr>
<td>3-fold</td>
<td>wastes more data than 10-fold, more expensive than validation set</td>
<td>slightly better than validation-set</td>
</tr>
<tr>
<td>N-fold</td>
<td></td>
<td>Identical to Leave-one-out</td>
</tr>
</tbody>
</table>
CV-based Model Selection

- We’re trying to decide which algorithm to use.
- We train each machine and make a table...

<table>
<thead>
<tr>
<th>$f_i$</th>
<th>Training Error</th>
<th>10-FOLD-CV Error</th>
<th>Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_3$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_4$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_5$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_6$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CV-based Model Selection

- Example: Choosing “k” for a k-nearest-neighbor regression.
- Step 1: Compute LOOCV error for six different model classes:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Training Error</th>
<th>10-fold-CV Error</th>
<th>Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>k=1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k=2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k=3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k=4</td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td>k=5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k=6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Step 2: Choose model that gave best CV score
- Train it with all the data, and that’s the final model you’ll use
CV-based Model Selection

• Why stop at $k=6$?
  • No good reason, except it looked like things were getting worse as $K$ was increasing

• Are we guaranteed that a local optimum of $K$ vs LOOCV will be the global optimum?
  • No, in fact the relationship can be very bumpy

• What should we do if we are depressed at the expense of doing LOOCV for $k = 1$ through 1000?
  • Try: $k=1, 2, 4, 8, 16, 32, 64, \ldots, 1024$
  • Then do hillclimbing from an initial guess at $k$
Cross Validation Notes

- After we chose non-tunable parameters/classifiers, retrain chosen classifier on all training data

<table>
<thead>
<tr>
<th>Training</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td></td>
<td>20%</td>
</tr>
<tr>
<td>Training</td>
<td>Validation</td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td></td>
<td>20%</td>
</tr>
<tr>
<td>Validation</td>
<td>Training</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>20%</td>
</tr>
</tbody>
</table>

- Based on 4-fold cross validation error, chose $T = 200$ for AdaBoost
- Retrain AdaBoost classifier with $T = 200$ on all Training data now:

<table>
<thead>
<tr>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20%</td>
</tr>
</tbody>
</table>
Cross Validation Notes

- Should still have separate Test set, not touched during cross-validation

- Sometime report just CV-results, no separate Test data

- Common practice, but should be aware that extensive use of CV could overfit to the data