Lecture 7: Model Complexity Trade-Offs and K-Fold Cross Validation

Mat Kallada

STAT2450 - Introduction to Data Mining
Just a Few Preliminary Notes

Assignment 1 is due tonight at 11:59pm

Assignment 2 will be released later tonight as well!

E-mail me your submission at kallada@cs.dal.ca
FAQ: Can I use code snippets I find online?

Yes - please cite sources however; some assignments will require you to do research. Google is your friend!

Use a comment-block to cite the source.

```c
// The function to detect letters was found at: stackoverflow.com/questions/14964960/select2-s
```
FAQ: Can I use code snippets I find online?

It’s important that you understand the logical principles of programming.

In the final exam, we will ask you questions about key programming concepts.
Outline for Today

Decision Tree Review ←
Trip-ups in Supervised Data Mining
K-Fold Cross Validation
Grid Search for Parameter Selection
Decision trees: A Brief Review

Builds a **tree-based model** for making predictions

“Decisions” or “Splits” correspond to questions about each feature

“End Leaves” correspond to predictions

Generates something like this

To predict, **pass-down** the unknown observation.
Decision trees: How are they built?

Built by repeatedly making “best split” out of all possible splits.

**Classification Trees**

\[
\text{argmax} \quad \text{Classification Accuracy of Split Position}
\]

**Regression Trees**

\[
\text{argmin} \quad \text{Mean Absolute Error of Split Position}
\]
Decision trees: How are they built?

Automatically generated by repeatedly making single splits.

It will keep on splitting until all the branches are composed of one class or value.
Decision Trees: We can re-use the same decision tree
During the prediction phase, we can re-use the same decision tree

This is why decision trees are faster to make predictions with than KNN.

We can’t “re-use” anything with KNN during prediction times
## Training Phases for Decision Trees vs KNN

<table>
<thead>
<tr>
<th>Training Phase (i.e. “train” in R)</th>
<th>Training Phase (i.e. “train” in R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build the Decision Tree on the Training Data</td>
<td>Nothing re-usable is made here</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predict Phase (for a new unknown observation)</th>
<th>Predict Phase (for a new unknown observation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass down the unknown observation in the decision tree from training phase</td>
<td>Compute distances to the unknown observation, Find K smallest Distances, Ect.</td>
</tr>
</tbody>
</table>
Decision trees: Visualization Abilities

The actual predictive process easily be visualized

It’s an actual tree after all.

How can I interpret predictive process of KNN? Not as easy.

Used heavily in medical applications

We can visualize the entire predictive process

There are no unexpected predictions.
Nick Bostrom’s take on Decision Trees

Oxford professor of Philosophy Nick Bostrom says decision trees are more favorable method when building robots due to predictive transparency.

He argues they allow understanding of decision making capabilities of intelligent agents.

Especially in case of retaliation.
Decision-Trees are a “white box” data mining method

Building a self-driving car, four classes: Drive Left, Drive Right, Accelerate, Brake

Features are things about the environment: “is_car_in_front”, “is_car_on_left_side”, ...

Allows us to understand how predictions are actually made!
Which is better: **Small** Trees or **Large** Trees

This is like the choice of K in KNN: which would perform better in the **real-world**?
Decision Trees: Complexity of the Tree

“Large” trees with a lot of branches will overfit the data.

“Small” trees with little branches underfit the data.

Sick - noise was avoided in smaller tree.
Decision Trees: *Too small decision trees don’t work*

The one below needs to be a little larger to capture the patterns.
Large Trees face Overfitting

Large trees fit onto every noisy point in our training data

And don’t work well in the real-world.

Large trees are like KNN with $K=1$ in terms of complexity.

But, trees that are too small don’t capture any patterns and underfit
Large Trees face **Overfitting**

Right here was a good spot to stop growing the tree

This one distinguishes pretty well between classes
Small Trees face **Underfitting**

This decision tree is **not complex enough**
An Idea: Let’s limit how large the tree can be

Maybe we could specify the maximum depth of the tree?

Maybe we could cut it off at a certain point.

Decision Trees by default make big trees.
Two ways we can prevent overfitting in decision trees

Use the "max_depth" parameter to specify the maximum depth that the tree can be.

Or use the "cp" parameter to limit the complexity. This reduces branches of the decision tree after being trained.

"cp" means make the split if it improves at least this much accuracy.

Smaller cp means larger decision tree.
Two ways we can prevent overfitting in decision trees
Specify them in the tuneGrid options in R

```r
rpart_options = data.frame( cp = 0.01, max_depth = 6)
rpart_model = train(x=animals[,1:3], y=animals[,4], method = "rpart", tuneGrid = rpart_options)
print("We just built predictive models with KNN and Decision Trees on the same data set!")
```

We’ll look at this more in detail next class!
**Remember**: KNN and Decision Trees solve the same problem

The **supervised data mining problem**: predicting a label of a dataset.

They just have **different approaches** to building $f(x)$.
**KNN vs Decision Trees**: Which one has **better** accuracy?

It depends on how your dataset is organized.

Some datasets KNN will work better

In other datasets Decision Trees works better

**Try them both** and see which performs better on a test set.

Use the model that performs better
Homework Tonight: Look through this Article

Reality check

Of course, what matters more is how the tree performs on previously-unseen data.

To test the tree's performance on new data, we need to apply it to data points that it has never seen before. This previously unused data is called test data.

Link: http://www.r2d3.us/visual-intro-to-machine-learning-part-1/
Outline for Today

Decision Tree Review

Trip-ups in Data Mining ←

K-Fold Cross Validation

Grid Search for Parameter Selection
Trip ups in Supervised Data Mining #1: Leaking the Test Set

Test set is used to estimate how well it performs in the real-world.

Never ever train on the test set

This is a type of “data leakage”
Trip ups in Supervised Data Mining #2: Imbalanced Classes

Someone builds a predictive model on binary classification for a rare disease.

Collects features of 15 patients with the disease and labels them “True”

Collects features of 1,000 patients without the disease and labels them “False”

He says he gets 99% accuracy on a test set.

Wait - is there something wrong here?
Trip ups in Supervised Data Mining #2: Imbalanced Classes

Someone builds a predictive model on binary classification for a rare disease.

Collects features of 15 patients with the disease and labels them “True”

Collects features of 1,000 patients without the disease and labels them “False”

He says he gets 99% accuracy on a test set.

Is there something wrong here.
Trip ups in Supervised Data Mining #2: Imbalanced Classes

Not a good assessment of model performance.

Some people **duplicate rows** of minority class to “re-balance” the data set.

Or **remove rows** of the majority class to re-balance them.
Trip ups in Supervised Data Mining #3: Concept Drift

After building your predictive model and deploying it, it gradually loses accuracy.

The actual underlying function changed over time.

This was an issue when PayPal first used data mining to predict fraud and having smart, “adaptive” fraudsters.
Trip ups in Supervised Data Mining #3: Concept Drift

We need to re-train our predictive model periodically on fresh data

This will ensure that it doesn’t go out of date
Outline for Today

Decision Tree Review

Trip-ups in Supervised Data Mining

K-Fold Cross Validation ←

Grid Search for Parameter Selection
Which one to use **Decision Trees** or **KNN**?

Someone gives you a dataset.

You’re ask to build a predictive model for classification.

What do you do?
Which one to use **Decision Trees** or **KNN**?

You cut-off a **set for testing** and another **set for training**

Run **KNN** and **Decision Tree** on Training Part

Check which one performs better on Testing Part.

Report the Testing Scores to your client.
We test the performance of both KNN and DT

We’ll just use the one that performs the best in the real-world!
You get *something* like this...

<table>
<thead>
<tr>
<th>Learning Method</th>
<th>Testing Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Trees(max_depth=10, cp=0.1)</td>
<td>86.4%</td>
</tr>
<tr>
<td>3-Nearest Neighbours</td>
<td>84.2%</td>
</tr>
<tr>
<td>15-Nearest Neighbours</td>
<td>83.8%</td>
</tr>
</tbody>
</table>

So - we should use decision trees right? It did better on the test set

Is there something wrong here?
**Issues with Hold-out Validation**

Hold-out validation “holds out” some rows just to test a model on. What if the test data was poorly chosen?

Our test dataset rows were randomly selected. They could be “easy” testing rows.
Issues with Hold-out Validation

Consider the dataset for addition...

<table>
<thead>
<tr>
<th>Number 1</th>
<th>Number 2</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>32139</td>
<td>321893821</td>
<td>321925960</td>
</tr>
<tr>
<td>9999</td>
<td>32823</td>
<td>42822</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Issues with Hold-out Validation

Consider the dataset for addition...

<table>
<thead>
<tr>
<th>Number 1</th>
<th>Number 2</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>32139</td>
<td>321893821</td>
<td>321925960</td>
</tr>
<tr>
<td>9999</td>
<td>32823</td>
<td>42822</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

You might get a model that has **zero error**, but we don’t know if it works =(.
K-Fold Cross Validation: A better assessment method

K-Fold Cross Validation is a method which computes a generalization statistic.

Determines how well a particular data mining method is on a data set.

Gives more robust generalization measure than the Hold-out Validation.
K-Fold Cross Validation: A better assessment method

1. Cut the data into K even parts (called “folds”)
2. Choose one of the parts for testing
3. Choose the other ones for training the model
4. Repeat this process K times for all possible test sets
5. Afterwards, average the performance on all test sets
# Simple Example of 5-Fold Cross Validation

Step 1. Cut up the Data into 5 parts (or 5 folds)

<table>
<thead>
<tr>
<th>Fold 1</th>
<th>Fold 2</th>
<th>Fold 3</th>
<th>Fold 4</th>
<th>Fold 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number 1</td>
<td>Number 2</td>
<td>Sum</td>
<td>Number 1</td>
<td>Number 2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>32139</td>
</tr>
<tr>
<td>32139</td>
<td>321893821</td>
<td>321925960</td>
<td>9999</td>
<td>32823</td>
</tr>
</tbody>
</table>

Okay - easy because there are only 5 rows

If there were 10 rows, each fold would have 2 rows in “5-Fold Cross Validation”
Simple Example of 5-Fold Cross Validation

Pick one fold for testing. Use the others for building a predictive model.

<table>
<thead>
<tr>
<th>Number 1</th>
<th>Number 2</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>32139</td>
<td>321893821</td>
<td>321925960</td>
</tr>
<tr>
<td>9999</td>
<td>32823</td>
<td>42822</td>
</tr>
</tbody>
</table>

So we build a predictive model, f(x), with Folds 2, 3, 4, 5.

Compute the Error (or accuracy) on the testing Fold 1.

Mean-Square Error: 4.2
Simple Example of 5-Fold Cross Validation

Pick one fold for testing. Use the others for building a predictive model.

<table>
<thead>
<tr>
<th></th>
<th>Number 1</th>
<th>Number 2</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fold 1 (TRAIN)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Fold 2 (TEST)</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Fold 3 (TRAIN)</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Fold 4 (TRAIN)</td>
<td>32139</td>
<td>321893821</td>
<td>321925960</td>
</tr>
<tr>
<td>Part 5 (TRAIN)</td>
<td>9999</td>
<td>32823</td>
<td>42822</td>
</tr>
</tbody>
</table>

So we build a predictive model, f(x), with Folds 1, 3, 4, 5.

Compute the Error (or accuracy) on the testing Fold 2.

Mean-Square Error: 0.12
Simple Example of 5-Fold Cross Validation

Pick one fold for testing. Use the others for building a predictive model.

<table>
<thead>
<tr>
<th>Fold 1 (TRAIN)</th>
<th>Number 1</th>
<th>Number 2</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>32139</td>
<td>321893821</td>
<td>321925960</td>
<td></td>
</tr>
<tr>
<td>9999</td>
<td>32823</td>
<td>42822</td>
<td></td>
</tr>
</tbody>
</table>

Mean-Square Error: 1.33

So we build a predictive model, f(x), with Folds 1, 2, 4, 5.

Compute the Error (or accuracy) on the testing Fold 3.
Simple Example of 5-Fold Cross Validation

Pick one fold for testing. Use the others for building a predictive model.

<table>
<thead>
<tr>
<th>Fold 1 (TRAIN)</th>
<th>Fold 2 (TRAIN)</th>
<th>Fold 3 (TRAIN)</th>
<th>Fold 4 (TEST)</th>
<th>Part 5 (TRAIN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number 1</td>
<td>Number 2</td>
<td>Sum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32139</td>
<td>321893821</td>
<td>321925960</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9999</td>
<td>32823</td>
<td>42822</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mean-Square Error: 44.2

So we build a predictive model, f(x), with Folds 1, 2, 3, 5.

Compute the Error (or accuracy) on the testing Fold 4.
Simple Example of 5-Fold Cross Validation

Pick one fold for testing. Use the others for building a predictive model.

<table>
<thead>
<tr>
<th>Number 1</th>
<th>Number 2</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>32139</td>
<td>321893821</td>
<td>321925960</td>
</tr>
<tr>
<td>9999</td>
<td>32823</td>
<td>42822</td>
</tr>
</tbody>
</table>

So we build a predictive model, f(x), with Folds 1, 2, 3, 4.

Compute the Error (or accuracy) on the testing Fold 5.

Mean-Square Error: 455.2
Compute the 5-Fold Cross-Validation Score

Now average those test scores up of those 5 predictive models:

\[
\frac{455.2 + 1.33 + 44.2 + 0.12 + 4.2}{5} = 101.01
\]

This data mining method has a 5-fold cross-validation score of 101.01

This is a more robust score than validating on only one of the parts
2-Fold Cross Validation

2-Fold Cross-validation Estimate: \((89+76)/2 = 82.5\%\)
3-Fold Cross Validation

[Diagram showing the process of 3-fold cross validation, including raw data, training and testing phases, and average score.]
K-Fold Cross Validation

All rows are eventually used for testing

So - there's no issue of using only "easy" test questions

Pick the method that has the best cross-validation estimate
How to pick K in K-Fold Cross Validation?

A lot of people either use K equals 10

These are generally accepted K values for comparing data mining methods

<table>
<thead>
<tr>
<th>Supervised Data Mining Method</th>
<th>10-Fold Cross Validation Classification Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree (max_depth=3)</td>
<td>85.56 +/- 4.2 %</td>
</tr>
<tr>
<td>10-KNN</td>
<td>83.63 +/- 1.3%</td>
</tr>
</tbody>
</table>
After determining the method you want

<table>
<thead>
<tr>
<th>Supervised Data Mining Method</th>
<th>10-Fold Cross Validation Classification Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree (max_depth=3, cp=0.001)</td>
<td>85.56 +/- 4.2 %</td>
</tr>
<tr>
<td>10-KNN</td>
<td>83.63 +/- 1.3%</td>
</tr>
</tbody>
</table>

Let’s use this one

Train with the entire dataset using the data mining method to build the final classifier
Outline for Today

Decision Tree Review

Trip-ups in Supervised Data Mining

K-Fold Cross Validation

Grid Search for Parameter Selection ←
Finding the **sweet spot** for model complexity

Some values of “cp” or K in KNN will give the best performance.

How complex should our model be?

![Diagram showing the relationship between model complexity and prediction score, with an optimal model complexity point in the middle. The gray curve represents the testing score, while the black curve represents the training score.](image)
External Parameters are Called **Hyperparameters**

These are values you specify beforehand.

Like K in K-nearest Neighbours

cp and max_depth in Decision Trees
Grid Search: Finding the Optimal External Values

Define a “grid” that you want to search over.

**K-Nearest Neighbours**

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>1, 2, 3, 4, 5, 6</td>
</tr>
</tbody>
</table>

**Decision Trees**

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>cp</td>
<td>0.01, 0.001, 0.003, 0.005</td>
</tr>
<tr>
<td>max_depth</td>
<td>4, 6, 8, 15, 16</td>
</tr>
</tbody>
</table>

Try all possible combinations and **pick the one with best CV score**.
print(“That’s all for today”)

We’ll talk more about grid searching next class.

Assignment 1 is due tonight at 11:59pm Atlantic Time.

Readings Posted on the Website

Assignment 2 will be posted shortly