CSC411/2515 Tutorial: K-NN and Decision Tree

Mengye Ren
csc{411,2515}ta@cs.toronto.edu

September 25, 2016
Cross-validation

$K$-nearest-neighbours

Decision Trees
Review: Motivation for Validation

Framework: learning as optimization

Goal: optimize model complexity (for our task)

Formulation: minimize underfitting and overfitting
Review: Motivation for Validation

Framework: learning as optimization

Goal: optimize model complexity (for our task)

Formulation: minimize underfitting and overfitting

In particular, we want our model to generalize well without overfitting.
Review: Motivation for Validation

Framework: learning as optimization

Goal: optimize model complexity (for our task)

Formulation: minimize underfitting and overfitting

In particular, we want our model to generalize well without overfitting.

We can ensure this by validating the model.
Types of Validation (1)

hold-out validation: split data into training set and validation set

▶ usually: 30% as hold-out set
Types of Validation (1)

**hold-out validation:** split data into training set and validation set

- usually: 30% as hold-out set

![Diagram of hold-out validation]

**Problems:**

- waste of dataset
- estimation of error rate may be misleading
Types of Validation (2)

cross-validation: random sub-sampling

1

Figure from Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
Types of Validation (2)

cross-validation: random sub-sampling

Problem:

▶ more computationally expensive than hold-out validation

---

1 Figure from Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
Variants of Cross-validation (1)

leave-$p$-out: use $p$ examples as the validation set, and the rest as training; repeat for all configurations of examples.

e.g., for $p = 1$: 

![Diagram showing leave-1-out cross-validation](image-url)
Variants of Cross-validation (1)

**leave-\(p\)-out:** use \(p\) examples as the validation set, and the rest as training; repeat for all configurations of examples.

E.g., for \(p = 1\):

Problem:

- **exhaustive:** We are required to train and test \(\binom{N}{p}\) times, where \(N\) is the number of training examples.
Variants of Cross-validation (2)

**K-fold:** partition training data into $K$ equally sized subsamples; for each fold, use $K - 1$ subsamples as training data with the last subsample as validation.
Variants of Cross-validation (2): $K$-fold

Advantages:

- All observations are used for both training and validation, and each observation is used for validation exactly once.
- non-exhaustive $\implies$ more tractable than LpOCV
Variants of Cross-validation (2): $K$-fold

Advantages:

- All observations are used for both training and validation, and each observation is used for validation exactly once.
- non-exhaustive $\implies$ more tractable than LpOCV

Problems:

- expensive for large $N$, $K$ (since we train/test $K$ models on $N$ examples)
  - but there are some efficient hacks to save time over the brute-force method . . .
- can still overfit if we validate too many models!
  - **Solution**: hold out an additional test set before doing any model selection, and check that the best model performs well even on the additional test set (*nested cross-validation*)
Practical Tips for Using $K$-fold Cross-validation

- **Q:** How many folds do we need?
- **A:** with larger $K$, ...
  - error estimation tends to be more accurate
  - but computation time will be greater

In practice:
- usually choose $K \approx 10$
- BUT larger dataset = choose smaller $K$
Practical Tips for Using $K$-fold Cross-validation

- **Q:** How many folds do we need?

- **A:** with larger $K$, ...
  - error estimation tends to be more accurate
  - but computation time will be greater

**In practice:**

- usually choose $K \approx 10$
- BUT larger dataset $\implies$ choose smaller $K$
Cross-validation

$K$-nearest-neighbours

Decision Trees
**K-nearest-neighbours: Definition**

**Training:** store all training examples (perfect memory)

**Test:** predict value/class of an unseen (test) instance based on closeness to stored training examples, relative to some distance (similarity) measure

\(^2\)Figure from Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT press.
Predicting with $K$-nearest-neighbours

- for $K = 1$,
  - predict the same value/class as the nearest instance in the training set.

- for $K > 1$,
  - find the $K$ closest training examples, and either
    - predict class by *majority vote* (in classification).
    - predict value by *average weighted inverse distance* (in regression).
Practical Tips for Using $K$NN (1)

- Ties may occur in a classification problem when $K > 1$
  - For binary classification: choose $K$ odd to avoid ties
  - For multi-class classification:
    - Decrease the value of $K$ until the tie is broken
    - If that doesn’t work, use the class given by a 1NN classifier
Practical Tips for Using KNN (2)

- magnitude of $K$:
  - smaller $K$: predictions have higher variance (less stable)
  - larger $K$: predictions have higher bias (less true)
Aside: The Bias-variance Tradeoff

A learning procedure creates **biased** models if . . .

- the predictive distribution of the models differs greatly from the target distribution.
Aside: The Bias-variance Tradeoff

A learning procedure creates biased models if . . .

- the predictive distribution of the models differs greatly from the target distribution.

A learning procedure creates models with high variance if . . .

- the models have greatly different test predictions (across different training sets from the same target distribution).
Practical Tips for Using $K$NN (2)

- magnitude of $K$:
  - smaller $K$: predictions have higher variance (less stable)
  - larger $K$: predictions have higher bias (less true)

Cross-validation can help here!
Practical Tips for Using KNN (3)

- the choice of distance measure affects the results!
  - e.g., standard Euclidean distance:

\[ d_E(x,y) = \sqrt{\sum_i (x_i - y_i)^2} \]

Problems:

- assumes features have equal variance
Practical Tips for Using \textit{KNN} (3)

- the choice of distance measure affects the results!
  - e.g., standard Euclidean distance:
    \[
    d_E(x, y) = \sqrt{\sum_i (x_i - y_i)^2}
    \]

Problems:

- assumes features have equal variance
  - \textbf{Solution}: standardize / scale the feature values

- assumes features are uncorrelated
Practical Tips for Using KNN (3)

- the choice of distance measure affects the results!
  - e.g., standard Euclidean distance:

\[
    d_E(x, y) = \sqrt{\sum_i (x_i - y_i)^2}
\]

Problems:

- assumes features have equal variance
  - **Solution**: standardize / scale the feature values

- assumes features are uncorrelated
  - **Solution**: use a more complex model (e.g., a Gaussian)
Practical Tips for Using KNN (3)

▶ the choice of distance measure affects the results!
  ▶ e.g., standard Euclidean distance:

\[ d_E(x, y) = \sqrt{\sum_i (x_i - y_i)^2} \]

Problems:

▶ assumes features have equal variance
  ▶ **Solution**: standardize / scale the feature values

▶ assumes features are uncorrelated
  ▶ **Solution**: use a more complex model (e.g., a Gaussian)

▶ in high-dimensional space, noisy features dominate
Practical Tips for Using KNN (3)

- the choice of distance measure affects the results!
  - e.g., standard Euclidean distance:
    \[ d_E(x, y) = \sqrt{\sum_i (x_i - y_i)^2} \]

Problems:

- assumes features have equal variance
  - **Solution**: standardize / scale the feature values

- assumes features are uncorrelated
  - **Solution**: use a more complex model (e.g., a Gaussian)

- in high-dimensional space, noisy features dominate
  - **Solution**: apply (learn?) feature weightings
Practical Tips for Using KNN (4)

- KNN has perfect memory, so computational complexity is an issue
  - at test time: $\mathcal{O}(N \cdot D)$ computations per test point

- Solutions:
  - dimensionality reduction
  - sample features
  - project the data to a lower dimensional space
  - sample training examples
  - use clever data structures, like $k$-D trees
Practical Tips for Using KNN (4)

- KNN has perfect memory, so computational complexity is an issue
  - at test time: $O(N \cdot D)$ computations per test point

Solutions:

- dimensionality reduction
  - sample features
    - project the data to a lower dimensional space
  - sample training examples
- use clever data structures, like k-D trees
MATLAB Demo
Cross-validation

\(K\)-nearest-neighbours

Decision Trees
Decision Trees: Definition

Goal: Approximate a discrete-valued target function

Representation: a tree, of which

- each internal (non-leaf) node tests an attribute
- each branch corresponds to an attribute value
- each leaf node assigns a class

---

Decision Trees: Induction

The ID3 algorithm:

- while training examples are not perfectly classified, do
  - choose the “most informative” attribute $\theta$ (that has not already been used) as the decision attribute for the next node $N$ (greedy selection)
  - for each value (discrete $\theta$) / range (continuous $\theta$), create a new descendant of $N$
  - sort the training examples to the descendants of $N$
### Decision Trees: Example *PlayTennis*

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
After splitting the training examples first on *Outlook* . . .

What should we choose as the next attribute under the branch *Outlook* = Sunny?
Choosing the “Most Informative” Attribute

Formulation: Maximise information gain over attributes $Y$.

Information Gain ($PlayTennis \mid Y$)

\[ = H(PlayTennis) - H(PlayTennis \mid Y) \]

\[ = \sum_x P(PlayTennis = x) \log P(PlayTennis = x) \]

\[ - \sum_{x,y} P(PlayTennis = x, Y = y) \log \frac{P(Y = y)}{P(PlayTennis = x, Y = y)} \]
Information Gain Computation (1)

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

\[
\text{InfoGain (} \text{PlayTennis} \mid \text{Humidity} \) = 0.970 - \frac{3}{5}(0.0) - \frac{2}{5}(0.0) = 0.970
\]
**Information Gain Computation (2)**

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

\[
\text{InfoGain (} \text{PlayTennis} \mid \text{Temperature}) = 0.970 - \frac{2}{5}(0.0) - \frac{2}{5}(1.0) - \frac{1}{5}(0.0) = 0.570
\]
Information Gain Computation (3)

\[
\text{InfoGain (PlayTennis | Wind)} = 0.970 - \frac{2}{5}(1.0) - \frac{3}{5}(0.918) = 0.019
\]
The Decision Tree for *PlayTennis*
Recall: The Bias-variance Tradeoff

- Where do decision trees naturally lie in this space?
- Answer: high variance
- How to fix: pruning (e.g., reduced-error pruning, rule post-pruning)
Recall: The Bias-variance Tradeoff

Where do decision trees naturally lie in this space?

- **Answer:** high variance
Recall: The Bias-variance Tradeoff

Where do decision trees naturally lie in this space?

**Answer**: high variance

**How to fix**: pruning (e.g., reduced-error pruning, rule post-pruning)