Learning from Data

Russell and Norvig Chapter 18

Learning

- Essential for agents working in unknown environments
- Learning is useful as a system construction method
  - Expose the agent to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms to improve performance

Examples of learning tasks

- OCR (Optical Character Recognition)
- Loan risk diagnosis
- Medical diagnosis
- Credit card fraud detection
- Speech recognition (e.g., in automatic call handling systems)
- Spam filtering
- Collaborative filtering (recommender systems)
- Biometric identification (fingerprints, iris scan, face)
- Information retrieval (incl. web searching)
- Data mining, e.g. customer purchase behavior
- Customer retention
- Bioinformatics: prediction of properties of genes and proteins.

Learning from examples

- Supervised learning: Given labeled examples of each digit, learn a classification rule

Learning

- The agent tries to learn from the data (examples) provided to it.
- The agent receives feedback that tells it how well it is doing.
- There are several learning scenarios according to the type of feedback:
  - Supervised learning: correct answers for each example
  - Semi-Supervised learning: correct answers for some examples
  - Unsupervised learning: correct answers not given
  - Reinforcement learning: occasional rewards (e.g. learning to play a game).
- Each scenario has appropriate learning algorithms: e.g. classification algorithms for supervised learning, clustering for unsupervised learning.
Learning

- The agent tries to learn from the data (examples) provided to it.
- The agent receives feedback that tells it how well it is doing.
- There are several learning scenarios according to the type of feedback:
  - Supervised learning: correct answers for each example
  - Semi-Supervised learning: correct answers for some examples
  - Unsupervised learning: correct answers not given
  - Reinforcement learning: occasional rewards (e.g., learning to play a game).
- Where do Bayesian networks fit in?

Unsupervised learning

- We are given \( x_1, \ldots, x_n \)
- Want to learn something "useful" about the data.
- What might we want to learn?

Supervised learning

Learning from examples – a.k.a. inductive learning

*Given a target function, \( f \)*

*An example is a pair \((x, f(x))\)*

Problem: find a hypothesis \( h \) such that \( h = f \) on a training set of examples

training set: \((x_1, f(x_1)), \ldots, (x_n, f(x_n))\)

Inductive learning tasks

- Classification: the desired outputs belong to a discrete set (Yes/No, 0,1,\ldots,9)
- Regression: the desired outputs are real numbers

Inductive learning

- Construct \( h \) to agree with \( f \) on training set
- Example: curve fitting
Inductive learning

- Construct $h$ to agree with $f$ on training set
- Example: curve fitting

Ockham’s razor: prefer the simplest hypothesis consistent with data

Learning is concerned with accurate prediction of future data, not accurate prediction of training data.

Supervised Learning

Example: want to classify

Data: Labeled images

$$D = \{(x_i, y_i)\}_{i=1}^n$$

$y_i \in \{\text{monkey, human}\}$

$x_i$ is a vector that represents the image

Task: Here is a new image: monkey or human?

The Nearest Neighbor Method

(your first classification algorithm)

$\text{NN(image)}$: 
1. Find the image in the training data which is closest to the query image.
2. Return its label.

Distance measures

- How to measure closeness?

query

closest image
Distance measures

- How to measure closeness?
- Discrete data: Hamming distance
- Continuous data: Euclidean distance
- Sequence data: edit distance
- Alternative: use a similarity measure (or dot product) rather than a distance

k-NN

- Use the closest k neighbors to make a decision instead of a single nearest neighbor
- Why do you expect this to work better?

Remarks on NN methods

- Very easy to implement
- No training required. All the computation performed in classifying an example (complexity: \(O(n)\))
- Need to store the whole training set (memory inefficient).
- Flexible, not many prior assumptions (a type of non-parametric classifier; does not assume anything about the data).
- k-NN variants robust against "bad examples".
- Curse of dimensionality: if data has many features that are irrelevant/noisy distances are always large.

Take home question

- How would you convert the k-nearest-neighbor classification method to a regression method?

Measuring classifier performance

- Or how accurate is my classifier.

Measuring classifier performance

The error rate on a set of examples \(D = \{(x_i, y_i)\}_{i=1}^n\):

\[
\frac{1}{n} \sum_{i=1}^{n} I(f(x_i) \neq y_i)
\]

\(I\) is the indicator function that returns 1 if its argument is True and 0 otherwise.

- What is the error rate of a nearest neighbor classifier applied to its training set?
Measuring classifier performance

- The error rate on a set of examples
  \[ \mathcal{D} = \{(x_i, y_i)\}_{i=1}^n \] :
  \[ \frac{1}{n} \sum_{i=1}^n I(f(x_i) \neq y_i) \]
  where \( I \) is the indicator function that returns 1 if its argument is True and zero otherwise.

- Report error rates computed on an independent test set (classifier was trained using training set):
  classifier performance on the training set is not indicative of performance on unseen data.

- Issue when classes are imbalanced.
- There are other measures of performance: area under ROC curves, area under precision-recall curves.

In practice:

- Split data into training set and test set (say 70%, 30%).
- Compare several classifiers trained on this split.
- Train final classifier on the full dataset.

A better method: cross-validation

Cross-validation

- Split data into \( k \) parts (\( E_1, \ldots, E_k \)) for \( i = 1, \ldots, k \):
  - training set = \( \mathcal{D} \setminus E_i \)
  - test set = \( E_i \)
  - classifier.train(training set)
  - accumulate results of classifier.test(test set)

This is called \( k \)-fold cross-validation
- Extreme version: Leave-One-Out
- Assumptions?

Uses of CV

Cross Validation is used to choose:
- Classifier parameters
  - \( k \) for k-NN
- Normalization method
- Which classifier
- Feature selection (which features provide best performance).
- This is called model selection

CV-based model selection

We're trying to determine which classifier to use

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Training error</th>
<th>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 the k-NN algorithm:

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Training error</th>
<th>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>

The general workflow

- Get data
- Decide on a representation (what features to use)
- Choose a classifier
- Assess the performance of the classifier
- Depending on the results: modify the representation, classifier, or look for more data

Next

- More classifiers:
  - Decision trees
  - How to use a probabilistic model such as a Bayesian network as a classifier