INTRODUCTION TO ARTIFICIAL INTELLIGENCE

EPISODE 7: MACHINE LEARNING
TODAY’S MENU

1. WHY MACHINE LEARNING?
2. KINDS OF ML
3. NEAREST NEIGHBORS
4. NAIVE BAYES (AGAIN!)
WHY MACHINE LEARNING?
WHAT IS MACHINE LEARNING?

• Definition:
  – machine = computer, computer program (in this course)
  – learning = improving performance on a given task, based on experience / examples

• In other words, instead of the programmer writing explicit rules for how to solve a given problem, the programmer instructs the computer how to learn from examples

• In many cases the computer program can even become better at the task than the programmer is!
KINDS OF MACHINE LEARNING

• Supervised machine learning:
  – task is to predict the correct (or good) response $y$ given an input $x$, e.g.:
    + classify samples to normal and abnormal
    + classify emails as spam or legit ("ham")
    + predict movie profits based on director, actors, ...
    + generate text descriptions of images

• Unsupervised machine learning:
  – task is to create models or summaries of the input $x$ (no $y$):
    + clustering (users, products, text documents by topic, ...)
    + building dependency graphs (Bayesian networks, ...)
    + reducing dimensionality to the essentials
    + visualization (dimension reduction to 2D/3D)
UNSUPERVISED LEARNING

• Clustering: form subsets of similar items
• Dimension reduction: represent data using a small number of features
• Visualization

(GRAPH) CLUSTERING
KINDS OF MACHINE LEARNING (CONTINUED)

• **Reinforcement learning:**
  – supervised learning but no direct feedback about the goodness of individual choices; instead delayed reward/penalty
    + e.g., win/lose a game,
    + reach destination successfully/not,
    + ...

• **Semi-supervised learning:**
  – supervised learning task but only some training data is labeled

• We'll mostly focus on **supervised learning**
SUPERVISED LEARNING

• Given input X, predict output/response Y
  – Classification: discrete Y
  – Regression: continuous Y
SUPERVISED LEARNING

• Example: **Face recognition**

  ![Images of faces](image1.png)

  ↓ ↓ ↓ ↓ ↓

  patrik antti doris patrik ...

• **Manual solution:** Program rules such as IF big nose AND usually eyeglasses AND short, dark hair THEN Mikko, ...

• **ML solution:** Use a large set of training images with known labels (person) to learn relevant features and construct a classifier.
NEAREST NEIGHBOR CLASSIFIER

• One of the simplest possible classifiers

1. Store all training data \((x_1, y_2), ..., (x_n, y_n)\)

2. Given a new test vector \(x^{(\text{test})}\), find the nearest training vector \(x^{(\text{NN})}\), and output its class label \(y^{(\text{NN})}\).
NEAREST NEIGHBOR CLASSIFIER

• One of the simplest possible classifiers

1. Store all training data \((x_1, y_2), \ldots, (x_n, y_n)\)
2. Given a new test vector \(x^{(\text{test})}\), find the nearest training vector \(x^{(\text{NN})}\), and output its class label \(y^{(\text{NN})}\).
NEAREST NEIGHBOR CLASSIFIER

• How to define "nearest"?

• For example, the distance between two binary vectors can be defined as the number of bits that differ:

\[
\begin{array}{cccccccccccc}
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
\hline
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0
\end{array}
\]

=> distance = 4

• If the bit vectors are B/W images, then this corresponds to comparing them pixel by pixel

• Is this a good distance measure? What happens when an image is shifted a couple of pixels to one direction?
NEAREST NEIGHBOR CLASSIFIER

- MNIST example (Exercises 4.4 & 4.5)

TRAINING DATA (KNOWN LABELS):

0 7 1 1 4 9 4 3 4 8 2 2 1 8 7 0 8 1 0 7
0 7 1 / 4 9 4 3 4 8 2 2 1 8 7 0 8 1 / 0 7

TEST DATA:
1

1, 7, 9, 1 / 8, 5, 7, 5, 0, 6, 6, 0, 4, 1, 2, 3, 4, 4
NEAREST NEIGHBOR CLASSIFIER

- MNIST example (Exercises 4.4 & 4.5)

TRAINING DATA (KNOWN LABELS):
0 7 1 1 4 9 4 3 4 8 2 2 1 8 7 0 8 1 0 7
0 7 1 1 4 9 4 3 4 8 2 2 1 8 7 0 8 1 0 7

TEST DATA:
1
0 7 9 1 8 5 7 5 0 6 6 0 4 1 2 3 4 4
NEAREST NEIGHBOR CLASSIFIER

- MNIST example (Exercises 4.4 & 4.5)

TRAINING DATA (KNOWN LABELS):

```
0 7 1 1 4 9 4 3 4 8 2 2 1 8 7 0 8 1 0 7
0 7 1 1 4 9 4 3 4 8 2 2 1 8 7 0 8 1 0 7
```

TEST DATA:

```
1 7 9 1 1 8 5 7 5 0 6 6 0 4 1 2 3 4 4
```

9
NEAREST NEIGHBOR CLASSIFIER

- MNIST example (Exercises 4.4 & 4.5)

TRAINING DATA (KNOWN LABELS):
0 7 1 1 4 9 4 3 4 8 2 2 1 8 7 0 8 1 0 7
0 7 1 1 4 9 4 3 4 8 2 2 1 8 7 0 8 1 0 7

TEST DATA:
1 7 9 1 1 8 5 7 5 0 6 6 0 4 1 2 3 4 4

- 7.6% error rate (5000 training data, 1000 test data)
K-NEAREST NEIGHBOR CLASSIFIER

- Similar to the NN classifier but take the k nearest points instead of just one
- Choose the majority label among them as the predicted label

K=3:

- training example of class 1
- training example of class 2
- test example
Treat each input feature (e.g., pixel) as a variable

Build a model $P(\text{feature}_i \mid \text{class})$ for each feature $i$

For B/W pixels, the probability $P(\text{pixel} = \text{'white'} \mid \text{class})$ can be visualized as a gray-scale value: the higher the probability the lighter the shade

This creates "prototypes" for each digit
NAIVE BAYES CLASSIFIER (AGAIN!)

\[\text{train\_NB(data)}:\]
1: for each class \(i\) in 0,...,9:
2: \hspace{1em} model.prior[\(i\)] = \#\{class=\(i\}\} / n
3: \hspace{1em} for each pixel \(j\) in 0,...,783:
4: \hspace{2em} model.cpt[\(i,j\)] = \#\{X_j=1, \text{class}=i\} / \#\{\text{class}=i\}
5: \hspace{1em} return model

\[\text{test(model, image)}:\]
1: \(Z = 0; \text{posterior} = \text{empty array}\)
2: for each class \(i\) in 0,...,9:
3: \hspace{1em} posterior[\(i\)] = model.p[\(i\)]
4: \hspace{1em} for each feature \(j\) in 0,...,783:
5: \hspace{2em} if image[\(j\)] = 1:
6: \hspace{3em} posterior[\(i\)] = posterior[\(i\)] * model.cpt[\(i,j\)]
7: \hspace{2em} else:
8: \hspace{3em} posterior[\(i\)] = posterior[\(i\)] * (1-model.cpt[\(i,j\)])
9: \hspace{1em} \(Z = Z + \text{posterior}[i]\)
10: for each class \(i\) in 0,...,9:
11: \hspace{1em} \text{posterior}[i] = \text{posterior}[i] / Z
12: return posterior

MNIST CASE
NEURAL NETWORK CLASSIFIERS

- Each node (neuron) computes a **weighted** sum of its inputs, applies a non-linear activation function, and passes it forward to the next layer.
- The output is given by the last layer.
- The **weights** are adapted to produce the correct output (supervised learning).
SUMMARY

• Machine learning is useful when:
  – the task is hard to solve manually (we can't solve the problem or it's hard to explain how we solve it)
  – we have access to data
  – the system needs to be able to adapt to different situations or users

• Start by defining
  – the task (what output should be produced),
  – goodness measure (classification error, win/loss, speed, ...)
  – data

• Then consider various alternative methods and test
MORE INFORMATION

• Machine learning, problem definitions, supervised/unsupervised learning:

   Introduction to Data Analysis using Machine Learning
   David Taylor (1h 8min), Youtube

• k-NN: Nearest Neighbor Classification
  Charles Elkan (pdf)

• Wikipedia: Naive Bayes classifier
MORE INFORMATION

- Introduction to Machine Learning (period 2)
- Advanced Machine Learning
- Deep Learning (period 2)
- ...
...
MORE INFORMATION

SUPER CRUNCHERS
HOW ANYTHING CAN BE PREDICTED
IAN AYRES

THE MASTER ALGORITHM
HOW THE QUEST FOR THE ULTIMATE LEARNING MACHINE WILL REMAKE OUR WORLD
PEDRO DOMINGOS

“Pedro Domingos demystifies machine learning and shows how wondrous and exciting the future will be.” —WALTER ISAACSON