COMP111: Artificial Intelligence

Section 10. Learning

Frank Wolter

Content

- Introduction and Examples
- Two basic (but very popular) supervised learning algorithms:
  - Naive Bayes classifier
  - *k*-Nearest Neighbor classifier

What is Machine Learning?

- Learning is the process of converting experience into expertise or knowledge.
- The input to a learning algorithm is training data (representing experience) and the output is some expertise, which usually takes the form of another computer program that can perform some task.

Types of Learning

- Supervised Learning: the learner is presented with examples

$$(x_1, L(x_1)), \ldots, (x_n, L(x_n)) \in \mathcal{X} \times \mathcal{Y}$$

  of labelled data x_i ($L(x_i)$ is the label of x_i) and the task is to generalize the examples to a function $f : \mathcal{X} \to \mathcal{Y}$. If $\mathcal{Y}$ is finite, then f is also called a classifier.
- Unsupervised Learning: the learner aims to find structure (also called patterns) in data without using examples. Clustering data into similar ones is a typical unsupervised learning problem.
- Reinforcement Learning: learning from rewards or punishments in a dynamic setting in which an agent has a long-term goal (winning a game, driving a car). Different from supervised learning as actions of the learner are not directly classified.

We focus on supervised learning of classifiers.

Example 1: Hand-written digit recognition

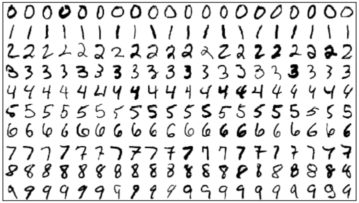


Figure: Images are 28×28 pixels

An input image is given as a vector $\vec{x} = (x_1, \ldots, x_{28\times28})$ of real numbers. Learn a classifier

$$f : \{ \text{ images } \} \to \{0,1,2,3,4,5,6,7,8,9\}$$

Hand-written digit recognition

- Assume we have training data:
  - hand-written digits with labels from $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- Then we have a supervised learning problem: generalise the training data to a classifier f.
- One of the first commercial applications of machine learning (ZIP codes).
- Take a look at MNIST database (`https://en.wikipedia.org/wiki/MNIST_database`).

Example 2: Face detection

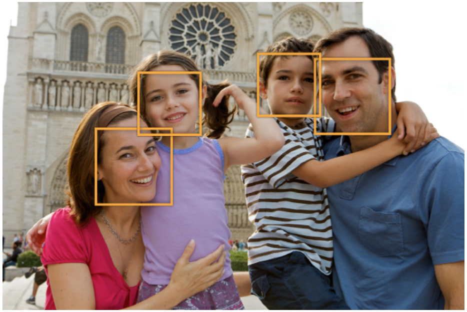


Figure: Image windows

Learn a classifier

$$f : \{ \text{ image windows } \} \mapsto \{\text{non-face, frontal face, profile-face}\}$$

Face detection

Figure: Labelled training data

Again a supervised learning problem:

- start with training data: image windows labelled with $\{$non-face, frontal face, profile-face$\}$
- generalise the training data to classifier f.

Example 3: Spam Detection

- Detect whether an incoming email is spam or not.
- A supervised learning problem: learn classifier

  $$f : \{\text{emails}\} \to \{\text{spam, not-spam}\}$$

  from labelled input data (emails).
- input data is word-count (e.g., word viagra in email indicates spam).
- requires a learning system as “enemy” keeps innovating.

When do we need machine learning?

- When do we need machine learning rather than directly program our computers to carry out the task at hand?
- Two aspects: the problem's complexity and the need for adaptivity.

Tasks that are too complex to program

- Tasks performed by animals/humans: There are numerous tasks that we human beings perform routinely, yet our introspection concerning how we do them is not sufficiently elaborate to extract a well defined program.
- We have seen handwritten digit recognition, face detection, spam detection, and so on.
- Further examples: driving, speech recognition, and playing games.
- State of the art machine learning programs achieve quite satisfactory results.

Tasks that are too complex to program

- Tasks beyond human capabilities: Another wide family of tasks that benefit from machine learning techniques are related to the analysis of very large and complex data sets.
- Examples: astronomical data, turning medical archives into medical knowledge, weather prediction, analysis of genomic data, Web search engines, and stock prediction.
- Meaningful information buried in data archives that are way too large and too complex for humans to make sense of.

Adaptivity

- One limiting feature of programmed tools is their rigidity – once the program has been written down and installed, it stays unchanged. However, many tasks change over time or from one user to another.
- Machine learning tools (programs whose behavior adapts to their input data) offer a solution to such issues; they are, by nature, adaptive to changes in the environment they interact with.
- Examples: decode handwritten text, where a fixed program can adapt to variations between the handwriting of different users; spam detection programs, adapting automatically to changes in the nature of spam e-mails; and speech recognition programs.

Supervised learning of classifiers

Given:

- A set $\mathcal{X}$ of possible instances to be classified (for example, emails, hand-written digits, image windows).
- A finite set $\mathcal{Y}$ of classes (for example, $\{$spam, not spam$\}$).
- Training data $(x_1, L(x_1)), \ldots, (x_n, L(x_n)) \in \mathcal{X} \times \mathcal{Y}$. $L(x_i)$ is called the label of x_i.
- A class of functions $\mathcal{F}$ from $\mathcal{X}$ to $\mathcal{Y}$ from which the classifier f is selected.

Aim:

- Compute classifier $f \in \mathcal{F}$ such that

$$f(x_i) \approx L(x_i)$$

for all training data $(x_i, L(x_i))$ such that for new $x \in \mathcal{X}$:

$$f(x) = y$$

is a good prediction for the class of x.

Towards the Naive Bayes Classifier

- One wants to learn a classifier $f : \mathcal{X} \to \mathcal{Y}$.
- we have training data

$$(x_1, L(x_1)), \ldots, (x_m, L(x_m))$$

- For a new $x \in \mathcal{X}$, the classifier predicts $f(x) = y \in \mathcal{Y}$ if

$$P(Y = y \mid x) \geq P(Y = y' \mid x)$$

  for all $y' \in \mathcal{Y} \setminus \{y\}$ (one takes the maximally probable hypothesis).
- Thus, in the derivation of the naive Bayes' classifier we introduce a random variable Y that takes values in $\mathcal{Y}$.

Towards the Naive Bayes Classifier

- We also make the (very common) assumption that every element of $\mathcal{X}$ is given by values $e_1, \ldots, e_n$ of features $E_1, \ldots, E_n$. We model those as random variables as well. Thus $P(Y = y \mid x)$ stands for

$$P(Y = y \mid E_1 = e_1, \ldots, E_n = e_n)$$

- Our training data then takes the form:

$$(e_1^1, \ldots, e_n^1, y_1), \ldots, (e_1^m, \ldots, e_n^m, y_m)$$

Example: Playing Tennis

- We assume we know the weather on a particular day, and want to predict whether Peter plays Tennis on that day.
- Also assume we have collected data on the weather on days on which Peters plays or does not play Tennis.
- The features used when collecting weather data are
  - the feature *outlook* with values: sunny, overcast, rain;
  - the feature *temp* with values: hot, mild, cool;
  - the feature *humidity* with values: high, normal;
  - the feature *wind* with values: weak, strong;

Example: Training data

Day	*outlook*	*temp*	*humidity*	*wind*	*play*
D1	sunny	hot	high	weak	No
D2	sunny	hot	high	strong	No
D3	overcast	hot	high	weak	Yes
D4	rain	mild	high	weak	Yes
D5	rain	cool	normal	weak	Yes
D6	rain	cool	normal	strong	No
D7	overcast	cool	normal	strong	Yes
D8	sunny	mild	high	weak	No
D9	sunny	cool	normal	weak	Yes
D10	rain	mild	normal	weak	Yes
D11	sunny	mild	normal	strong	Yes
D12	overcast	mild	high	strong	Yes
D13	overcast	hot	normal	weak	Yes
D14	sunny	mild	high	strong	No

Example: Prediction

Assume we want to predict whether Peter plays Tennnis on a day on which

- *outlook*=sunny;
- *temp*=cool;
- *humidity*=high;
- *wind*=strong.

In what follows we write

$$P(\text{sunny} \mid \text{Yes})$$

for

$$P(\mathit{outlook} = \text{sunny} \mid \mathit{PlaysTennis} = \text{Yes})$$

and so on for the remaining random variables and probabilities.

Example: Estimating probabilities

We would like to estimate:

$$P(Y = y \mid E_1 = e_1, \ldots, E_n = e_n)$$

which in the example is

$$P(\text{Yes} \mid \text{sunny}, \text{cool}, \text{high}, \text{strong})$$

But how can we do this directly if in the training data there is no entry for a day on which the outlook is sunny, the temperature is cool, the humidity is high, and the wind is strong?

Towards the Naive Bayes Classifier

By Bayes' Theorem

$$P(Y = y \mid E_1 = e_1, \ldots, E_n = e_n) = \frac{P(E_1 = e_1, \ldots, E_n = e_n \mid Y = y)P(Y = y)}{P(E_1 = e_1, \ldots, E_n = e_n)}$$

As we only want to compare probabilities, it is sufficient to estimate

$$P(E_1 = e_1, \ldots, E_n = e_n \mid Y = y)P(Y = y)$$

for all $y \in \mathcal{Y}$.

Example: Estimating probabilities

Thus, we want to estimate:

$$P(\text{sunny}, \text{cool}, \text{high}, \text{strong} \mid \text{Yes})P(\text{Yes})$$

and

$$P(\text{sunny}, \text{cool}, \text{high}, \text{strong} \mid \text{No})P(\text{No})$$

We can estimate $P(\text{Yes})$ by dividing the number of days on which Peter plays tennis by the total number of days; and we can estimate $P(\text{No})$ by dividing the number of days on which Peter does not play tennis by the total number of days.

But as there are no entries for days on which the outlook is sunny, the temperature is cool, the humidity is high, and the wind is strong, how can we estimate $P(\text{sunny}, \text{cool}, \text{high}, \text{strong} \mid \text{Yes})$?

Example: Assume conditional independence

Then we can "estimate"

$$P(\text{sunny}, \text{cool}, \text{high}, \text{strong} \mid \text{Yes})P(\text{Yes})$$

by

$$P(\text{sunny}|\text{Yes})P(\text{cool} \mid \text{Yes})P(\text{high} \mid \text{Yes})P(\text{strong} \mid \text{Yes})P(\text{Yes})$$

and

$$P(\text{sunny}, \text{cool}, \text{high}, \text{strong} \mid \text{No})P(\text{No})$$

by

$$P(\text{sunny}|\text{No})P(\text{cool} \mid \text{No})P(\text{high} \mid \text{No})P(\text{strong} \mid \text{No})P(\text{No})$$

Probabilities such as $P(\text{sunny}|\text{Yes})$ can be estimated by dividing the number of days on which Peter plays tennis and on which it is sunny by the total number of days on which Peter plays tennis.

Example: Prediction

Using the training data we estimate:

$$V_{\text{Yes}} = P(\text{Yes})P(\text{sunny} \mid \text{Yes})P(\text{cool} \mid \text{Yes})P(\text{high} \mid \text{Yes})P(\text{strong} \mid \text{Yes})$$

and

$$V_{\text{No}} = P(\text{No})P(\text{sunny} \mid \text{No})P(\text{cool} \mid \text{No})P(\text{high} \mid \text{No})P(\text{strong} \mid \text{No})$$

Example: Prediction

We estimate, for example,

- $P(\text{Yes}) = \frac{9}{14}$;
- $P(\text{No}) = \frac{5}{14}$
- $P(\text{strong} \mid \text{Yes}) = \frac{3}{9}$.

and obtain:

$$V_{\text{Yes}} = 0.0053, \quad V_{\text{No}} = 0.0274$$

Thus, the naive Bayes' classifier predicts that Peter will not play tennis.

Naive Bayes Classifier

Suppose we want to predict the class of the instance with features $(e_1, \ldots, e_n)$.
We assume $E_1, \ldots, E_n$ are independent given Y and estimate

$$V_y = P(Y = y) \prod_{i=1}^{n} P(E_i = e_i \mid Y = y)$$

for all $y \in \mathcal{Y}$ as follows:

- $P(Y = y)$ is estimated by the number of instances labelled with y in the training data divided by the number of all instances in the training data;
- $P(E_i = e_i \mid Y = y)$ is estimated by the number of instances with feature e_i labelled as y in the training data divided by the number of all instances labelled with y in the training data.

We then set

$$f(e_1, \ldots, e_n) = y$$

for the y for which V_y is maximal (take the maximally probable hypothesis).

Example: Spam filter

- One wants to learn a classifier

$$f : \{\text{emails}\} \to \{\text{spam, not spam}\}$$

  Thus, in this case

$$\mathcal{X} = \{ \text{ emails } \}, \quad \mathcal{Y} = \{\text{spam, not spam}\}$$

- We have training data:

$$(\text{email}_1, \text{spam}), (\text{email}_2, \text{not spam}), \ldots$$

Representation of emails

- We represent every email x by the words from the English dictionary that occur in it. Assume the English dictionary has 50.000 words and take an enumeration of these words. Then we introduce random variables $E_1, \ldots, E_{50.000}$ which take values in $\{0, 1\}$, where
  - $(E_i = 1)$ says that word number i occurs in the email.
  - $(E_i = 0)$ says that word number i does not occur in the email.
- The email x we want to classify is then given by the sequence

$$e_1 \cdots e_{50000} \in \{0, 1\}^{50.000}$$

  stating which words occur in x.

Classification

- To estimate

$$V_{\text{Yes}} = P(\text{Spam} = 1) \prod_{i=1}^{50000} P(E_i = e_i \mid \text{Spam} = 1)$$

  and

$$V_{\text{No}} = P(\text{Spam} = 0) \prod_{i=1}^{50000} P(E_i = e_i \mid \text{Spam} = 0)$$

  - estimate $P(\text{Spam} = 1)$ by the number of spam emails divided by the total number of emails in the training data.
  - estimate $P(E_i = 1 \mid \text{Spam} = 1)$ by the number of spam emails containing word number i divided by the number of all spam emails in the training data.
  - estimate $P(E_i = 0 \mid \text{Spam} = 0)$ by the number of non spam emails not containing word number i divided by the number of non-spam emails in the training data.
- Classify the email as spam if $V_{\text{Yes}} \geq V_{\text{No}}$.

Summary: Naive Bayes Classifier

- A supervised learning algorithms based on Bayes Theorem;
- It is called "naive" because it is assumed that the features are independent of each other, given the classification;
- Naive Bayes classifier works surprizingly well in practice even if the features are obviously not independent given the classification.

Similarity

- Assume that we can measure the similarity between items in $\mathcal{X}$ to be classified.
- For example, one could measure how similar two emails are by counting the number of words from the English dictionary they both contain and compare it to the number of words only one of the two emails contains.

Idea. To classify a new data item x, a similarity-based classifier considers the classification of the items most similar to x in the training data and classifies x accordingly.

Distance Measures

We measure similarity between data items using a distance measure d which assigns to data items x, x' a non-negative number

$$d(x, x') \in \mathbb{R}.$$

$d(x, x')$ is called the distance between x and x'.

One typically requires that d satisfies the following equations for metric spaces:

- $d(x, y) = 0$ if and only if $x = y$ (identity of indiscernibles);
- $d(x, y) = d(y, x)$ (symmetry);
- $d(x, y) + d(y, z) \geq d(x, z)$ (triangle inequality).

Distances for data items with numerical features

Numerous distance measures have been introduced. Of particular importance is the Euclidean distance which gives, in the two and three dimensional case, the *length of the straight line between points*:

- The Euclidean Distance

  $$d((e_1, \ldots, e_n), (f_1, \ldots, f_n))$$

  between $(e_1, \ldots, e_n)$ and $(f_1, \ldots, f_n)$ is

  $$\sqrt{\sum_{i=1}^{n} (e_i - f_i)^2}$$

- for $n = 1$, $d(e, f) = \sqrt{(e - f)^2} = |e - f|$.
- for $n = 2$,

  $$d((e_1, e_2), (f_1, f_2)) = \sqrt{(e_1 - f_1)^2 + (e_2 - f_2)^2}$$

Distances for data with non-numerical features

Consider qualitative data such as:

Day	*outlook*	*temp*	*humidity*	*wind*	*play*
D1	sunny	hot	high	weak	No
D2	sunny	hot	high	strong	No
D3	overcast	hot	high	weak	Yes
D4	rain	mild	high	weak	Yes

What should the distance between D1 and D2 (as far as the weather is concerned) be?

A natural distance measure in this case is the matching distance between values of features: the number of features on which Di and Dj do no coincide. Then

- $d(\mathrm{D1}, \mathrm{D2}) = 1$
- $d(\mathrm{D1}, \mathrm{D4}) = 2$

Nearest Neighbor Classifier

Assume the data in $\mathcal{X}$ come with a distance measure $d(x, x') \in \mathbb{R}$ which states how similar $x, x' \in \mathcal{X}$ are. Then the nearest neighbor classifier works as follows:

Given training data

$$(x_1, L(x_1)), \ldots, (x_n, L(x_n))$$

for a new $x \in \mathcal{X}$ to be classified, let x_i be the element of the training set that is nearest to x:

$$d(x, x_i) \leq d(x, x_j), \qquad \text{for all } x_j \neq x_i.$$

Then define the value $f(x)$ of the classifier f by setting

$$f(x) = L(x_i).$$

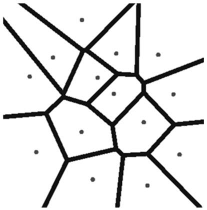
Nearest Neighbor Classifier: pseudocode

```
Input: training data $(x_0, L(x_0)), \ldots, (x_n, L(x_n))$
       new $x \in \mathcal{X}$ to be classified

for $i = 0$ to $n$ do
    Compute distance $d(x, x_i)$
endfor
return $L(x_i)$ such that $d(x, x_i)$ is minimal
```

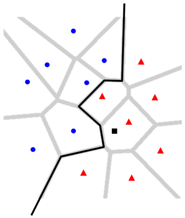
Nearest Neighbor Classifier

A nearest neighbor classifier partitions $\mathcal{X}$ into regions. Boundaries are equal distance from training points.



Nearest Neighbor Classifier

Thus, we obtain the following classifier for the classes “blue circle” and “red triangle”:



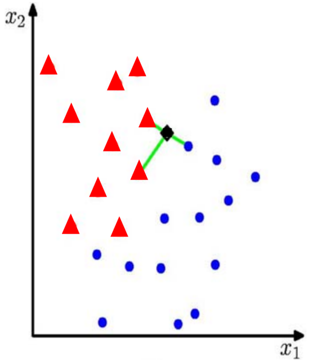
The Nearest Neighbor Classifier can be problematic

- The Nearest Neighbor Classifier is very close to training data.
- It does not generalize the training data: if the training data is noisy, then the classifier is noisy as well.
- For example, if a spam email is erroneously labelled as non-spam, then all similar emails will also be classified as non-spam.

k-Nearest Neighbor Classifier

- For x to be classified find k nearest $x_{i_1}, \dots, x_{i_k}$ in the training data.
- Classify x according to the majority vote of their class labels.

For $k = 3$:



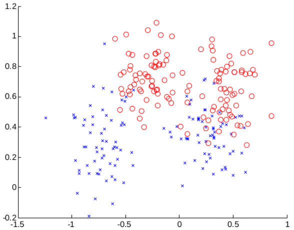
k-Nearest Neighbor Classifier

```
1: Input: training data $(x_0, L(x_0)), \dots, (x_n, L(x_n))$
2:        new $x \in \mathcal{X}$ to be classified
3:
4: for $i = 0$ to $n$ do
5:        Compute distance $d(x, x_i)$
6: endfor
7: Let $x_{i_1}, \dots, x_{i_k}$ be the list of items such that
8:        $d(x, x_{i_j})$ is among the $k$ smallest distances
9: return label $L$ which occurs most frequently in
   $L(x_{i_1}), \dots, L(x_{i_k})$ (majority vote)
```

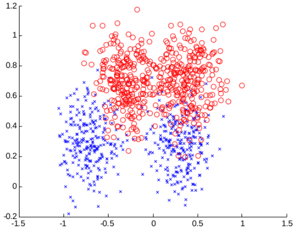
Learning a "good" k

To learn a "good" k divide labelled data into training data T and validation data V.

Training data



Validation data



Learning a “good” k

The classification error of a classifier f on a set of data items is the number of incorrectly classified data items divided by the total number of data items.

Let

- f_1 be the 1-nearest neighbor classifier obtained from the training data T;
- f_2 be the 2-nearest neighbor classifier obtained from the training data T;
- and so on for $3, \ldots, m$.

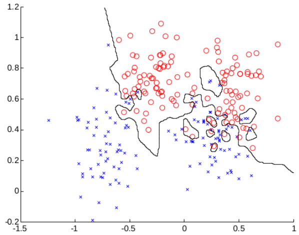
Choose f_k for which the classification error is minimal on the validation data.

Generalization

- The aim of supervised learning is to do well on test data that is not known during learning.
- Choosing the values for parameters (here k) that minimize the classification error on the training data is not necessarily the best policy.
- We want the learning algorithm to model true regularities in the data and ignore noise in the data.

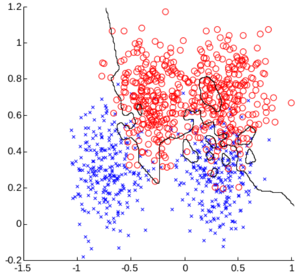
Training and validation data: $k = 1$

Training data



error = 0.0

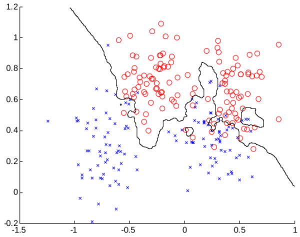
Validation data



error = 0.15

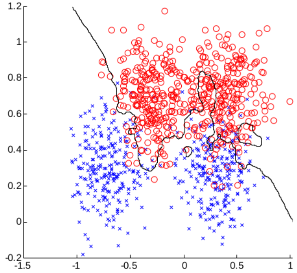
Training and validation data: $k = 3$

Training data



error $= 0.0760$

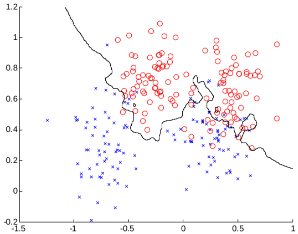
Validation data



error $= 0.1340$

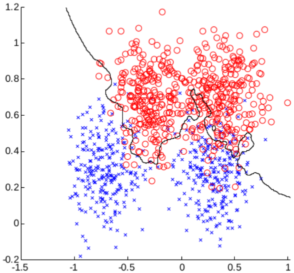
Training and validation data: $k = 7$

Training data



error $= 0.1320$

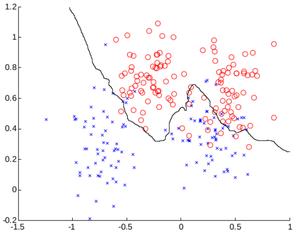
Validation data



error $= 0.1110$

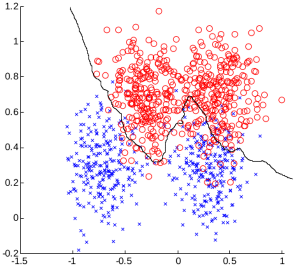
Training and validation data: $k = 21$

Training data



error = 0.1120

Validation data



error = 0.0920

Properties and training

As k increases:

- Classification boundary becomes smoother (possibly reflecting regularity in the data)
- Error on training data can increase.

Summary

Advantages:

- k-nearest neighbor is simple but effective
- Decision surfaces can be non-linear
- Only a single parameter, k, easily learned by cross validation.

Disadvantages:

- What does nearest mean? Need to specify a distance measure.
- Computational cost: must store and search through the entire training set at test time.