Introduction to Supervised Classification

Christian Hardmeier
(based on materials by Joakim Nivre)

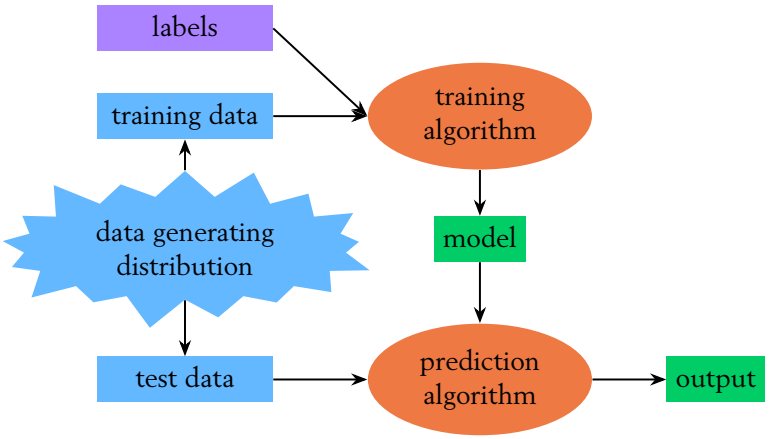
2018-04-03

Overview

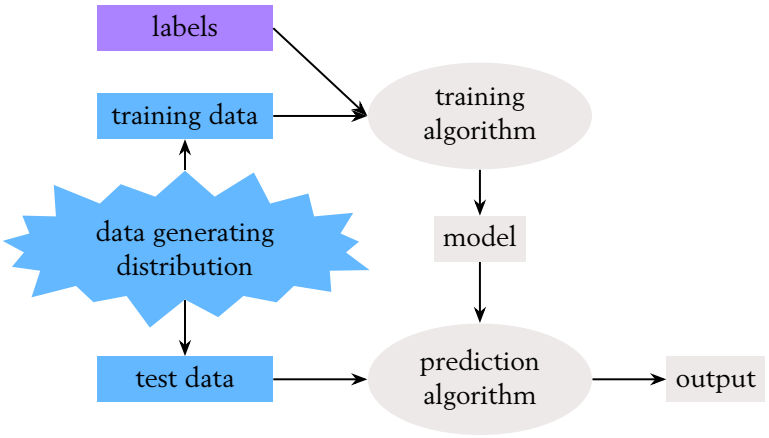
- Introducing basic concepts of classification
- Two intuitive classification algorithms:
  - Decision trees
  - *k*-nearest neighbours

Hard-coded approach

Supervised machine learning approach



Data



Data

- Training data and test data are assumed to come from the *same* data generating distribution.
- In supervised learning, we have labels for the training data.

$$T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

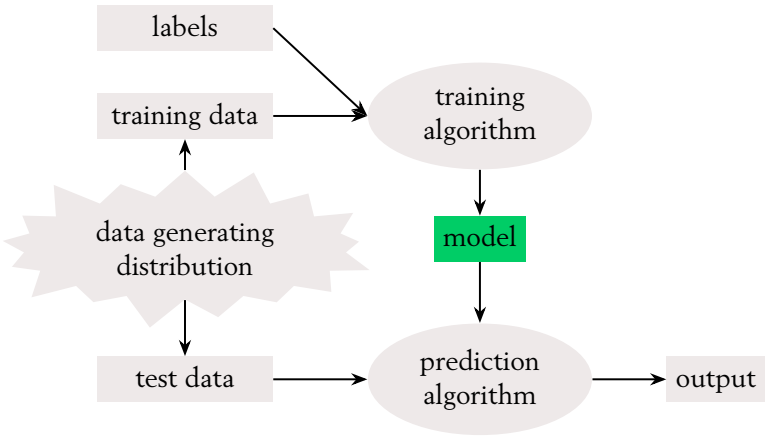
- Each input x is represented as a list (or vector) of features:

$$x = \langle \phi_1(x), \ldots, \phi_k(x) \rangle$$

Types of features

- A numerical feature has a real number as its value.
  - Examples: Word frequency, sentence length, similarity score. . .
  - Numerical features can be added, subtracted etc.
- A binary feature represents a yes/no distinction
  - Examples: Word starts with capital letter (or not), word is a name (or not).
  - Binary features are often encoded as 0 or 1.
- A categorical feature represents a choice between discrete alternatives.
  - Examples: Word forms, part-of-speech tags, word senses.
  - Categorical features can be converted to sets of binary features.

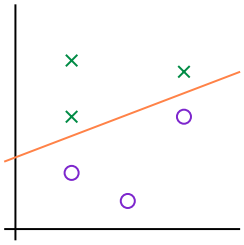
Model



Models, inductive bias and parameters

- The training algorithm creates a model to be used for prediction.
- The model consists of
  - fixed assumptions (inductive bias)
  - parameters that are tuned to the training data

Example: Linear classifier in 2D



Inductive bias:
Decision boundary is a straight line.

$$y = mx + c$$

Parameters:
slope m, intercept c

Parameters vs. hyperparameters

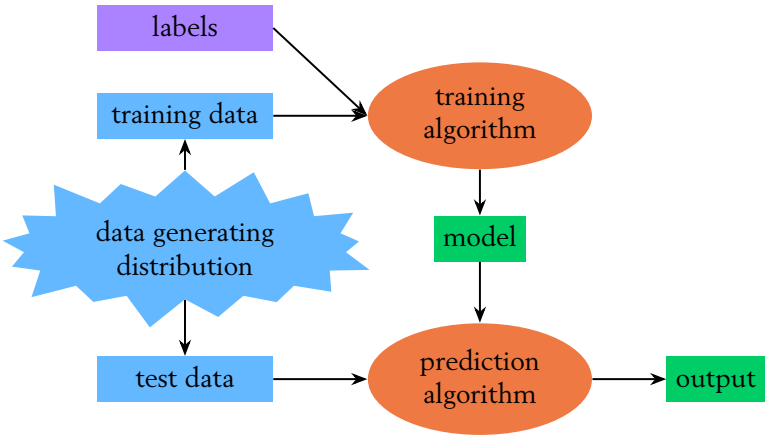
Parameters:

- Parameters are the tunable components of the model.
- Learnt from training data.
- Typical examples: probabilities, feature weights

Hyperparameters:

- Variables that control how parameters are learnt.
- Chosen a priori (or tuned using held-out data).
- Typical examples: model size (depth, complexity), learning rate

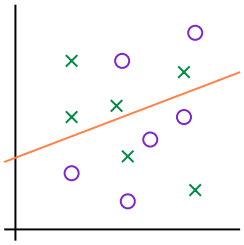
Supervised machine learning approach



Loss and expected loss

- The loss function measures how far off our predictions are from the truth.
- We want to find the model and parameters that minimise the expected loss under the data-generating distribution.
- As we don't know the true distribution, we try to minimise the loss on a training sample instead.

Choosing loss functions



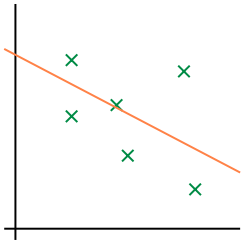
For classification:

- Zero-one loss:

$$\ell(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise} \end{cases}$$

- Weighted loss involving utility function

Choosing loss functions



For regression:

- Squared loss: $\ell(y, \hat{y}) = (y - \hat{y})^2$
- Absolute loss: $\ell(y, \hat{y}) = |y - \hat{y}|$

Underfitting and overfitting

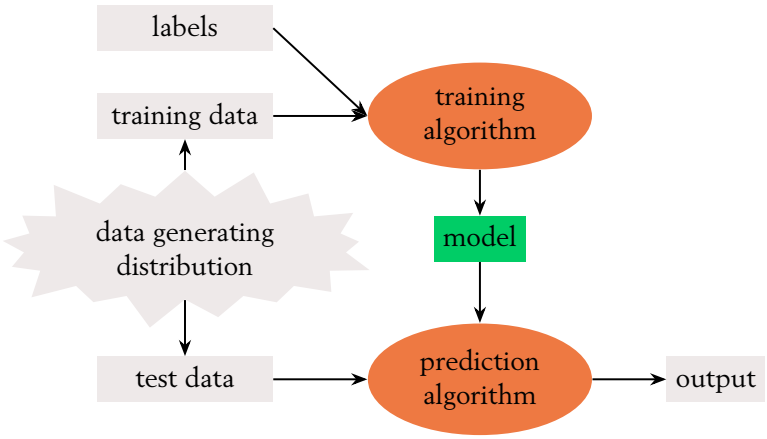
Underfitting:

- Learning algorithm is not sensitive enough to training data.
- The model doesn't learn.

Overfitting:

- Learning algorithm is too sensitive to training data.
- The model doesn't generalise.
- Classifier picks up noise in the training set.

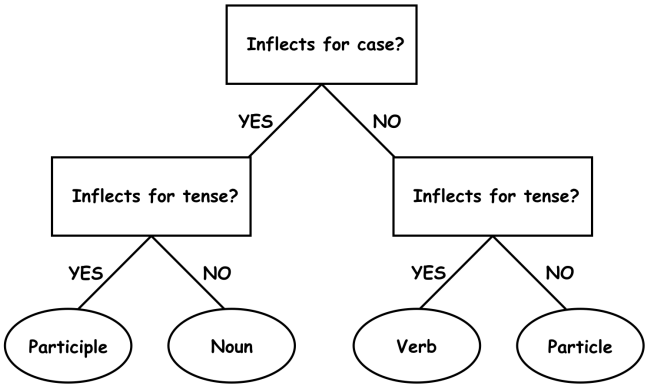
Classification methods



Decision trees

- Hierarchical tree structure for classification:
  - Each internal node specifies a test of some feature.
  - Each branch corresponds to a value of the tested feature.
  - Each leaf node provides a classification of the instance.
- Represents a *disjunction of conjunctions* of constraints
  - Each path from root to leaf specifies a conjunction of tests.
  - The tree itself represents the disjunction of all paths.

Varro's part-of-speech classification



Learning a decision tree

- Input:
  - Training set $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
  - Feature set $\Phi = \{\phi_1, \ldots, \phi_k\}$
- Base cases:
  - If $y_i = y$ for all i, create a leaf node with prediction y.
  - If Φ is empty, create a leaf node with *majority* prediction y.
- Recursive case:
  - Find the best feature $\phi \in \Phi$ and create a node labelled ϕ.
  - For every value v of ϕ:
    - Learn a subtree for $T_v = \{(x, y) | \phi(x) = v\}$ and $\Phi \backslash \{\phi\}$.
    - Create a branch labelled $\phi = v$ to this subtree.

How to select the best feature?

- Highest accuracy on training set:
  Which new feature would classify
  most of the remaining training instances correctly?
- Information gain:
  Approach based on information theory

Entropy

Entropy measures the difficulty of prediction:

$$H(Y) = -\sum_{y \in Y} P(y) \log_2 P(y)$$

Consider:

$$T = \{(x, 0), (x, 0), (x, 1), (x, 1)\} \qquad H(Y) = 1$$
$$T = \{(x, 0), (x, 0), (x, 0), (x, 1)\} \qquad H(Y) = 0.81$$
$$T = \{(x, 0), (x, 0), (x, 0), (x, 0)\} \qquad H(Y) = 0$$

Information gain

Information gain measures the reduction in entropy:

$$\begin{aligned} IG(Y, \phi) &= H(Y) - H(Y|\phi) \\ &= H(Y) - \sum_{v} P(\phi = v)\, H(Y_{\phi=v}) \end{aligned}$$

Consider:

$$T = \{(\langle 0, 0\rangle), 0), (\langle 0, 1\rangle, 0), (\langle 1, 0\rangle, 1), (\langle 1, 1\rangle, 1)\}$$

$T[\phi_1 = 0] = \{(\langle 0, 0\rangle, 0), (\langle 0, 1\rangle, 0)\}$ $\quad T[\phi_2 = 0] = \{(\langle 0, 0\rangle, 0), (\langle 1, 0\rangle, 1)\}$

$T[\phi_1 = 1] = \{(\langle 1, 0\rangle, 1), (\langle 1, 1\rangle, 1)\}$ $\quad T[\phi_2 = 1] = \{(\langle 0, 1\rangle, 1), (\langle 1, 1\rangle, 0)\}$

$IG(Y, \phi_1) = 1$ $\qquad IG(Y, \phi_2) = 0$

Pruning

- Decision trees are susceptible to overfitting.
- Remove subtrees for better generalisation:
  - Prepruning: Early stopping (entropy threshold)
  - Postpruning: Grow whole tree, prune subtrees using validation set
- Prepruning is faster, postpruning is more accurate.

Properties of decision trees

- Decision trees are appropriate for classification when
  - interpretation of learned model is important (rules).
  - features can be both categorical and numeric.
  - the data is naturally described in a disjunctive form.
  - training data may be noisy (missing values, incorrect labels).
- Inductive bias of (most) decision tree learners:
  - Prefers trees with informative attributes close to the root.
  - Prefers smaller trees over bigger ones (with pruning).
- Hyperparameters:
  - Split criterion (information gain, gain ratio)
  - Pruning method (tree depth, entropy threshold)

Nearest-neighbour classification

This "rule of nearest neighbor" has considerable elementary intuitive appeal and probably corresponds to practice in many situations. For example, it is possible that much medical diagnosis is influenced by the doctor's recollection of the subsequent history of an earlier patient whose symptoms resemble in some way those of the current patient.
(Fix and Hodges, 1952)

Key ideas:

- Storage of old instances
- Similarity-based reasoning to new instances

k-nearest neighbour classification

Learning:

- Store training instances $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

Classification:

- Given a new test instance x:
  - Compute a distance between x and each stored instance x_i.
  - Keep track of the k closest instances (nearest neighbours).
- Assign to x the majority prediction y of the k nearest neighbours.

Geometric justification:

- Proximity in feature space should imply similar prediction.
- Smoothness assumption.

Hyperparameters of a k-NN classifier

- Distance metric
  - How do we measure distance between instances?
  - Determines the layout of the instance space.
  - Votes of nearest neighbours can be weighted by distance.
  - You don't even need to represent the features explicitly to compute a distance (see CiML chapter 11)
- The value of k
  - How large a neighbourhood should we consider?
  - Determines the complexity of the hypothesis space.
  - Setting $k = 1$ may lead to overfitting.
  - Setting $k = N$ is just a majority baseline.
- Feature and instance weighting.

Properties of k-NN

- Nearest neighbour classification is appropriate when
  - the items in the training data are representative of the data to be classified.
  - training data may be noisy (missing values, incorrect labels).
  - Fast classification is not crucial.
- Inductive bias of k-NN:
  - Nearby instances should have the same label (smoothness).
  - All features are equally important (without feature weights).
- Hyperparameters:
  - Distance metric
  - The value of k
  - Weighting schemes for features and instances

Assignment 1

`http://stp.lingfil.uu.se/~ch/ml2018/lab1/ml-lab1.html`

- In the first assignment, you will explore decision trees and k-NN classifiers in a popular ML tool called WEKA.
- No lab session for this assignment, but please contact me if you get stuck.
- Submit report through Studentportalen by the 13th April.