ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická
Katedra kybernetiky

Learning and its types.

Petr Pošík

This lecture is based on the book
Ten Lectures on Statistical and Structural Pattern Recognition
by Michail I. Schlesinger and Václav Hlaváč (Kluwer, 2002).
(V české verzi kniha vyšla ve vydavatelství ČVUT v roce 1999 pod názvem
Deset přednášek z teorie statistického a strukturálního rozpoznávání).

Learning

Decision strategy design

Using an observation $x \in X$ of an object of interest with a hidden state $k \in K$, we should design a decision strategy $q : X \to D$ which would be optimal with respect to certain criterion.

Decision strategy design

Using an observation $x \in X$ of an object of interest with a hidden state $k \in K$, we should design a decision strategy $q : X \to D$ which would be optimal with respect to certain criterion.

Bayesian decision theory requires complete statistical information $p_{XK}(x, k)$ of the object of interest to be known, and a suitable penalty function $W : K \times D \to \mathcal{R}$ must be provided.

Non-Bayesian decision theory studies tasks for which some of the above information is not available.

In practical applications, typically, none of the probabilities are known! The designer is only provided with the **training (multi)set** $T = \{(x_1, k_1), (x_2, k_2), \ldots, (x_l, k_l)\}$ of examples.

- It is simpler to provide good examples than to gain complete or partial statistical model, build general theories, or create explicit descriptions of concepts (hidden states).
- The aim is to find definitions of concepts (classes, hidden states) which are
  - complete (all positive examples are satisfied), and
  - consistent (no negative examples are satisfied).
- The training (multi)set is *finite*, the found concept description is only a *hypothesis*.

Decision strategy design

Using an observation $x \in X$ of an object of interest with a hidden state $k \in K$, we should design a decision strategy $q : X \to D$ which would be optimal with respect to certain criterion.

Bayesian decision theory requires complete statistical information $p_{XK}(x,k)$ of the object of interest to be known, and a suitable penalty function $W : K \times D \to \mathcal{R}$ must be provided.

Non-Bayesian decision theory studies tasks for which some of the above information is not available.

In practical applications, typically, none of the probabilities are known! The designer is only provided with the **training (multi)set** $T = \{(x_1,k_1),(x_2,k_2),\ldots,(x_l,k_l)\}$ of examples.

- It is simpler to provide good examples than to gain complete or partial statistical model, build general theories, or create explicit descriptions of concepts (hidden states).
- The aim is to find definitions of concepts (classes, hidden states) which are
  - complete (all positive examples are satisfied), and
  - consistent (no negative examples are satisfied).
- The training (multi)set is *finite*, the found concept description is only a *hypothesis*.

When do we need to use learning?

- When knowledge about the recognized object is insufficient to solve the PR task.
- Most often, we have insufficient knowledge about $p_{X|K}(x|k)$.

Types of feedback in learning

Supervised learning:

- A training multi-set of examples is available. Correct answers (hidden state, class, the quantity we want to predict) are *known* for all observations.
  - **Classification:** the answers (the output variable of the model) are nominal, i.e. the value specifies a class ID. (predict spam/ham based on email contents, predict 0/1/.../9 based on the image of the number, etc.)
  - **Regression:** the answers (the output variable of the model) are quantitative, often continuous (predict temperature in Prague based on date and time, predict height of a person based on weight and gender, etc.)

Learning
- Strategy design
- Feedback
- Param. estimation
- Strategy selection
- Surrogate criteria
- Learning revisited
- Summary

Types of feedback in learning

Supervised learning:

- A training multi-set of examples is available. Correct answers (hidden state, class, the quantity we want to predict) are *known* for all observations.
  - **Classification:** the answers (the output variable of the model) are nominal, i.e. the value specifies a class ID. (predict spam/ham based on email contents, predict 0/1/. . . /9 based on the image of the number, etc.)
  - **Regression:** the answers (the output variable of the model) are quantitative, often continuous (predict temperature in Prague based on date and time, predict height of a person based on weight and gender, etc.)

Unsupervised learning:

- A training multi-set of examples is available. Correct answers are *not known*, they must be sought in data itself $\Rightarrow$ data analysis.

Types of feedback in learning

Supervised learning:

- A training multi-set of examples is available. Correct answers (hidden state, class, the quantity we want to predict) are *known* for all observations.
  - **Classification:** the answers (the output variable of the model) are nominal, i.e. the value specifies a class ID. (predict spam/ham based on email contents, predict 0/1/.../9 based on the image of the number, etc.)
  - **Regression:** the answers (the output variable of the model) are quantitative, often continuous (predict temperature in Prague based on date and time, predict height of a person based on weight and gender, etc.)

Unsupervised learning:

- A training multi-set of examples is available. Correct answers are *not known*, they must be sought in data itself $\Rightarrow$ data analysis.

Semisupervised learning:

- A training multi-set of examples is available. Correct answers are *known only for a subset* of the training set.

Types of feedback in learning

Supervised learning:

- A training multi-set of examples is available. Correct answers (hidden state, class, the quantity we want to predict) are *known* for all observations.
  - **Classification:** the answers (the output variable of the model) are nominal, i.e. the value specifies a class ID. (predict spam/ham based on email contents, predict 0/1/. . . /9 based on the image of the number, etc.)
  - **Regression:** the answers (the output variable of the model) are quantitative, often continuous (predict temperature in Prague based on date and time, predict height of a person based on weight and gender, etc.)

Unsupervised learning:

- A training multi-set of examples is available. Correct answers are *not known*, they must be sought in data itself $\Rightarrow$ data analysis.

Semisupervised learning:

- A training multi-set of examples is available. Correct answers are *known only for a subset* of the training set.

Reinforcement learning:

- A training multi-set of examples is *not available*. Correct answers, or rather rewards for good decisions in the past, *are given occasionally after decisions are taken*.

Learning as parameter estimation

1. **Assume** $p_{XK}(x,k)$ has a particular form (e.g. Gaussian, mixture of Gaussians, piece-wise constant) with a small number of parameters Θ_k.
2. **Estimate** the values of parameters Θ_k using the training set T.
3. **Solve** the classifier design problem as if the estimated $\hat{p}_{XK}(x,k)$ was the true (and unknown) $p_{XK}(x,k)$.

Learning as parameter estimation

1. **Assume** $p_{XK}(x,k)$ has a particular form (e.g. Gaussian, mixture of Gaussians, piece-wise constant) with a small number of parameters Θ_k.
2. **Estimate** the values of parameters Θ_k using the training set T.
3. **Solve** the classifier design problem as if the estimated $\hat{p}_{XK}(x,k)$ was the true (and unknown) $p_{XK}(x,k)$.

Pros and cons:

- If the true $p_{XK}(x,k)$ does not have the assumed form, the resulting strategy $q'(x)$ can be arbitrarilly bad, even if the training set size L approaches infinity.
- Implementation is often straightforward, especially if the parameters Θ_k are assumed to be independent for each class (**naive bayes classifier**).

Learning as optimal strategy selection

- Choose a class Q of strategies $q_\Theta : X \to D$. The class Q is usually given as a parametrized set of strategies of the same kind, i.e. $q_\Theta(x, \Theta_1, \ldots, \Theta_{|K|})$.
- The problem can be formulated as a non-Bayesian task with non-random interventions:
  - The unknown parameters Θ_k are the non-random interventions.
  - The probabilities $p_{X|K,\Theta}(x|k, \Theta_k)$ must be known.
  - The solution may be e.g. such a strategy that minimizes the maximal probability of incorrect decision over all Θ_k, i.e. strategy that minimizes the probability of incorrect decision in case of the worst possible parameter settings.
  - But even this minimal probability may not be low enough—this happens especially in cases when the class Q of strategies is too broad.
  - It is necessary to narrow the set of possible strategies using additional information—the training (multi)set T.
- **Learning** then amounts to **selecting a particular strategy q^*_Θ from the a priori known set** Q using the information provided as training set T.
  - Natural criterion for the selection of one particular strategy is the risk $R(q_\Theta)$, but it cannot be computed because $p_{XK}(x, k)$ is unknown.
  - The strategy $q^*_\Theta \in Q$ is chosen by minimizing some other surrogate criterion on the training set which approximates $R(q_\Theta)$.
  - The choice of the surrogate criterion determines the *learning paradigm*.

Several surrogate criteria

All the following surrogate criteria can be computed using the training data T.

Learning as parameter estimation

- according to the **maximum likelihood**.
- according to a **non-random training set**.

Learning as optimal strategy selection

- by **minimization of the empirical risk**.
- by **minimization of the structural risk**.

Several surrogate criteria

All the following surrogate criteria can be computed using the training data T.

Learning as parameter estimation

- according to the **maximum likelihood**.
  - The likelihood of an instance of the parameters $\Theta = (\Theta_k : k \in K)$ is the probability of T given Θ:

$$L(\Theta) = p(T|\Theta) = \prod_{(x_i,k_i)\in T} p_K(k_i) p_{X|K}(x_i|k_i, \Theta_{k_i})$$

  - Learning then means to find Θ^* that maximizes the probability of T:

$$\Theta^* = (\Theta^*_k : k \in K) = \arg\max_{\Theta} L(T, \Theta)$$

    which can be decomposed to

$$\Theta^*_k = \arg\max_{\Theta_k} \sum_{x \in X} \alpha(x,k) \log p_{X|K}(x|k, \Theta_k),$$

    where $\alpha(x,k)$ is the frequency of the pair (x,k) in T (i.e. T is multiset).
  - The recognition is then performed according to $q_\Theta(x, \Theta^*)$.
- according to a **non-random training set**.

Learning as optimal strategy selection

- by **minimization of the empirical risk**.
- by **minimization of the structural risk**.

Several surrogate criteria

All the following surrogate criteria can be computed using the training data T.

Learning as parameter estimation

- according to the **maximum likelihood**.
- according to a **non-random training set**.
  - When random examples are not easy to obtain, e.g. in recognition of images.
  - T is carefully crafted by the designer:
    - it should cover the whole recognized domain
    - the examples should be typical ("quite probable") prototypes
  - Let $T(k), k \in K$, be a subset of the training set T with examples for state k. Then

$$\Theta_k^* = \arg\max_{\Theta_k} \min_{x \in T(k)} p_{X|K}(x|k, \Theta_k)$$

  - Note that the Θ^* does not depend on the frequencies of (x, k) in T (i.e. T is a set).

Learning as optimal strategy selection

- by **minimization of the empirical risk**.
- by **minimization of the structural risk**.

Several surrogate criteria

All the following surrogate criteria can be computed using the training data T.

Learning as parameter estimation

- according to the **maximum likelihood**.
- according to a **non-random training set**.

Learning as optimal strategy selection

- by **minimization of the empirical risk**.
  - The set Q of parametrized strategies $q(x, \Theta)$, penalty function $W(k, d)$.
  - The quality of each strategy $q \in Q$ (i.e. the quality of each parameter set Θ) could be described by the risk

    $$R(\Theta) = R(q) = \sum_{k \in K} \sum_{x \in X} p_{XK}(x, k) W(k, q(x, \Theta)),$$

    but p_{XK} is unknown.
  - We thus use the *empirical risk* R_{emp} (training set error):

    $$R_{\text{emp}}(\Theta) = R_{\text{emp}}(q) = \frac{1}{|T|} \sum_{(x_i, k_i) \in T} W(k_i, q(x_i, \Theta)).$$

  - Strategy $q_\Theta(x, \Theta^*)$ is used where $\Theta^* = \arg\min_\Theta R_{\text{emp}}(\Theta)$.
  - Examples: Perceptron, neural networks (backprop.), classification trees, ...
- by **minimization of the structural risk**.

Several surrogate criteria

All the following surrogate criteria can be computed using the training data T.

Learning as parameter estimation

- according to the **maximum likelihood**.
- according to a **non-random training set**.

Learning as optimal strategy selection

- by **minimization of the empirical risk**.
- by **minimization of the structural risk**.
  - Based on Vapnik-Chervonenkis theory
  - Examples: Optimal separating hyperplane, support vector machine (SVM)

Learning revisited

Do we need learning? When?

- If we are about to solve one particular task which is sufficiently known to us, we should try to develop a recognition method *without learning*.
- If we are about to solve a task belonging to a well defined class (we only do not know which particular task from the class we shall solve), develop a recognition method *with learning*.

Learning revisited

Do we need learning? When?

- If we are about to solve one particular task which is sufficiently known to us, we should try to develop a recognition method *without learning*.
- If we are about to solve a task belonging to a well defined class (we only do not know which particular task from the class we shall solve), develop a recognition method *with learning*.

The designer

- should understand all the varieties of the task class, i.e.
- should find a solution to the whole class of problems.

Learning revisited

Do we need learning? When?

- If we are about to solve one particular task which is sufficiently known to us, we should try to develop a recognition method *without learning*.
- If we are about to solve a task belonging to a well defined class (we only do not know which particular task from the class we shall solve), develop a recognition method *with learning*.

The designer

- should understand all the varieties of the task class, i.e.
- should find a solution to the whole class of problems.

The solution

- is a parametrized strategy and
- its parameters are learned from the training (multi)set.

Learning revisited

Do we need learning? When?

- If we are about to solve one particular task which is sufficiently known to us, we should try to develop a recognition method *without learning*.
- If we are about to solve a task belonging to a well defined class (we only do not know which particular task from the class we shall solve), develop a recognition method *with learning*.

The designer

- should understand all the varieties of the task class, i.e.
- should find a solution to the whole class of problems.

The solution

- is a parametrized strategy and
- its parameters are learned from the training (multi)set.

The *supervised learning* is a topic for several upcoming lectures:

- Decision trees and decision rules.
- Linear classifiers.
- Adaboost.

Summary

Learning:

- Needed when we do not have sufficient statistical info for recognition.
- There are several types of learning differing in the types of information the learning process can use.

Approaches to learning:

- Assume p_{XK} has a certain form and use T to estimate its parameters.
- Assume the right strategy is in a particular set and use T to choose it.
- There are several learning paradigms depending on the choice of criterion used instead of Bayesian risk.