Probabilistic Graphical Models

Lecture Notes
Fall 2009

September 9, 2009

Byoung-Tak Zhang
School of Computer Science and Engineering &
Cognitive Science, Brain Science, and Bioinformatics
Seoul National University
http://bi.snu.ac.kr/~btzhang/

Chapter 1. Learning as a Lifelong Process

1.1 Learning Systems

Definition. A **learning system** is a natural or artificial system that improves its performance from experiential data through interaction with an environment.

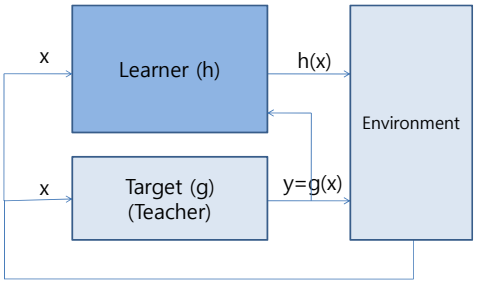

Figure 1-1. Learning system architecture

Learning as building a **model of the target system** from data

- **Target (goal) system**: $g: X \rightarrow Y, \quad y = g(x)$
- Input data $x_i \in X$
- Output data $y_i \in Y, \quad y_i = g(x_i)$
- Data set $D = \{(x_i, y_i) \mid i = 1,\ldots, n\}$
- **Model (hypothesis)** $h: X \rightarrow Y, \quad y = h(x), \quad h \in H$
- Learning algorithm $A: D \rightarrow H$

1.2 Learning as Mathematical Induction

Concept Learning

- X: Set of all possible instances x
- G: Set of target (goal) concepts g. $g : X \rightarrow \{0, 1\}$
- D: Set of positive and negative examples
- H: Set of possible hypotheses h $h : X \rightarrow \{0, 1\}$

- *L*: Learner: Outputs *h* in *H*, an estimate of *g*.

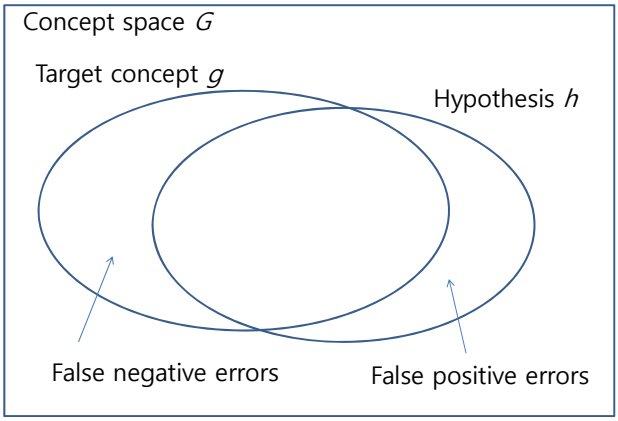

Figure 1-2. Learning as finding a hypothesis *h* for the target concept *g* in the concept space *G*.

Learning is to find an hypothesis *h* that best estimates the target concept *g* given a training set of positive and negative examples of the target concept. The (sample) error of the learner is measured by counting the number of mispredictions

$$E_D(h) = \frac{1}{|D|}\sum_{x \in D}[g(x) - h(x)]$$

The true error $E_D(h)$ of the learner can be measured by estimating the probability (with respect to *g* and *D*) that *h* will misclassify an instance drawn at random according to distribution of instances *D:*

$$E_D(h) = \Pr_{x \in D}[h(x) \neq g(x)]$$

Remark: $g(x)$ and $h(x)$ are functions that returns 0 or 1 in concept learning.

PAC Learning Framework

- Loosening the requirements of the learner
  (1) *Error* be bounded by some constant ε, that can be made arbitrarily small.
  (2) **Probability of failure** be bounded by some δ, that can be made arbitrarily small.
- Probably approximately correct (PAC) learning: The learner *probably* learns a hypothesis that is *approximately correct.* That is, the learner *L* will, with probability (1 - δ), output a hypothesis with maximum error ε.

$$\Pr(E_D(h) > \varepsilon) < \delta$$

- The target concept *G* is PAC-learnable by learner *L* using *H* if for all *g* in *G*, distributions *D* over *X*, 0 < ε < 1/2, 0 < δ < 1/2, learner *L* will with probability at least (1 - δ) output a hypothesis $h \in H$ such that $E_D(h) \leq \varepsilon$ in time that is polynomial in $1/\varepsilon$, $1/\delta$, n, $size(g)$.
  n: size of instances in *X*
  size(*g*): encoding length of *g* in *G*

Sample complexity for finite hypothesis spaces

- Deriving a bound on the number of training examples required by any consistent learner (outputs hypothesis that perfectly fit the training data), independent of the specific algorithm.
- Version space ($VS_{H,D}$): et of all hypotheses $h \in H$ that correctly classify the training examples *D*.

$$VS_{H,D} = \{h \in H \mid (\forall \langle x, g(x) \rangle \in D)(h(x) = g(x))\}$$

- To bound the number of examples needed by any consistent learner, we need only bound the number of examples needed to assure that the version space contains no unacceptable hypothesis.
- Exhausting the VS: The version space is said to be ε-exhausted with respect to g and D, if every hypothesis h in $VS_{H,D}$ has error less than ε with respect to g and D,

$$(\forall h \in VS_{H,D}) E_D(h) < \varepsilon$$

- All the hypotheses consistent with the observed training examples happen to have true error less than ε

- **Theorem: ε-exhausting the version space (Haussler 1988, from Mitchell 1997):** If H is finite, and D is a sequence of $m \geq 1$ examples, then for any $0 \leq \varepsilon \leq 1$, the probability that the version space $VS_{H,D}$ is not ε-exhausted is less than or equal to

$$|H| e^{-\varepsilon m}$$

- An upper bound on the probability that the version space is not ε-exhausted. This bounds the probability that m examples will fail to eliminate all bad hypotheses.
- **Sample complexity bound**. Reducing this probability below δ, we get a sample complexity bound:

$$|H| e^{-\varepsilon m} \leq \delta$$

$$m \geq \frac{1}{\varepsilon}(\ln |H| + \ln(1/\delta))$$

Example. PAC Learnability of Boolean Conjunctions

- Consider the class G of concepts described by conjunctions of boolean literals. There are three possibilities for each literal x_i: including x_i, including its negation, and ignoring it.
  The size of hypothesis space H is defined by conjunctions of literals based on n Boolean variables: $|H| = 3^n$
- Bound for the sample complexity

$$m \geq \frac{1}{\varepsilon}(n \ln 3 + \ln(1/\delta))$$

- Example: $n = 10$, 95% with error less than 0.1
  $M = 1/0.1 \times (10 \ln 3 + \ln(1/0.05)) = 140$
  It suffices to present 140 randomly drawn training examples.

- Conjunctions of Boolean literals are PAC-learnable
  m grows linearly in n, linearly in $1/\varepsilon$, and logarithmically in $1/\delta$.

Example. Unlearnability of k-term DNF (disjunctive normal form)
k-term DNF: $T_1 \vee T_2 \vee \ldots \vee T_k$ with T_i = conjunction of n literals

$$|H| = k3^n$$

$$m \geq \frac{1}{\varepsilon}(n \ln 3 + \ln k + \ln(1/\delta))$$

k-term DNF is *not* PAC learnable using $H=G$: Despite having *polynomial sample complexity*, the *computational complexity is not polynomial*, because this learning problem can be shown to be unsolvable in polynomial time.

Example. Unbiased Learners

- A set G of all definable target concepts corresponds to the power set of X: $|G| = 2^{|X|}$

$$|X| = 2^n$$

$$|G| = 2^{|X|} = 2^{2^n}$$

$$m \geq \frac{1}{\varepsilon}(2^n \ln 2 + \ln(1/\delta))$$

The unbiased class of target concepts has *exponential* sample complexity under the PAC model.

Sample Complexity for Infinite Hypothesis Spaces

- Another measure of the complexity of *H:* Vapnik-Chervonenkis dimension of *H*, or *VC*(*H*)
- VC dimension measures the complexity of *H*, not by the number of distinct hypotheses |*H*|, but by the number of distinct instances from *X* that can be completely discriminated using *H*. provides tighter bounds than |*H*| allows to characterize the sample complexity of many infinite hypothesis spaces.
- Shattering a set of instances

  A set of instances *S* is shattered by hypothesis space *H* if and only if for every dichotomy of *S* there exists some hypothesis in *H* consistent with this dichotomy.

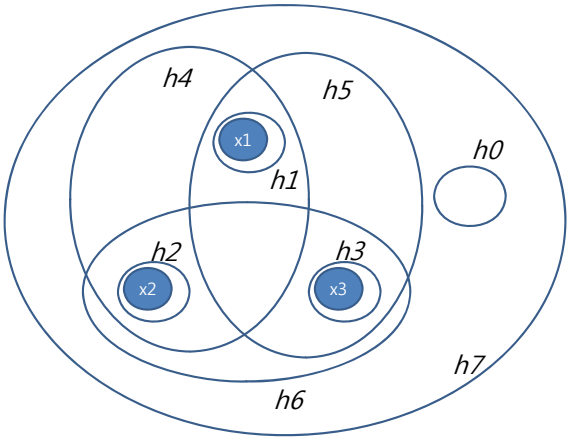

Figure 1-3. Shattering of the set of tree instances (x1, x2, x3) by 8 hypotheses (h0, h1, ..., h7).

VC Dimension

- The Vapnik-Chervonenkis dimension, *VC*(*H*), of hypothesis space *H* defined over instance space *X* is *the size of the largest finite subset of X shattered by H.*

  If arbitrarily large finite sets of *X* can be shattered by *H*, then $VC(H) \equiv \infty$.

  For any finite *H*,

  $$VC(H) \leq \log_2 |H|$$

  Suppose $VC(H) = d$. Then H will require 2^d hypos to shatter d inst..

  Hence, $2^d \leq |H|$, $d = VC(H) \leq \log_2 |H|$.

- **Definition.** A set of functions (a learning machine) has **VC dimension *d*** if there exist *d* samples that can be shattered but there do not exist *d+1* samples that can be shattered (maximum number of samples for which all possible binary labeling can be induced by the set of functions)
- **Example**. VC dimension of linear classifier over R^2 is 3

Sample Complexity and the VC Dimension

- Upper bound on sample complexity (Blumer et al. 1989)

$$m \geq \frac{1}{\varepsilon}(4\log_2(2/\delta) + 8VC(H)\log_2(13/\varepsilon))$$

Sufficient number for PAC learning any concept in G

- Lower bound on sample complexity (Ehrenfeucht et al. 1989)

$$\max\left[\frac{1}{\varepsilon}\log(1/\delta),\ \frac{VC(G)-1}{32\varepsilon}\right]$$

Necessary number for successful learning of any concept in G.

1.3 Learning as Statistical Approximation

Learning is considered as a function approximation problem based on noisy data.

Training data

$D = \{(\mathbf{x}_i, y_i) \mid y_i = g(\mathbf{x}_i),\ i = 1, ..., n\}$

$g(\cdot)$: target function

Approximating function

$f(\mathbf{x}, w)$ with parameter vector w

Loss function: measures the quality of approximation

- Loss: discrepancy between the output produced by the system and the learning machine for a given point x

$$L(y, f(\mathbf{x}, \omega))$$

- Example of loss: squared error

$$L(y, f(\mathbf{x}, \omega)) = [y - f(\mathbf{x}, \omega)]^2$$

Risk functional is the expected value of the loss

$$R(\omega) = \int L(y, f(\mathbf{x}, \omega))\, p(\mathbf{x}, y)\, d\mathbf{x}dy$$

- Learning is the process of estimating the function $f(\mathbf{x}, w_0)$ which minimizes the risk functional over the set of functions supported by the learning machine using the training data (the true $P(\mathbf{x}, y)$ is unknown).
  - Estimates $f(\mathbf{x}, w^*)$, the optimal solution obtained with finite data set using some learning procedure.

Inductive Principle

- How should a learning machine use training data?
- A general prescription for obtaining an estimate $f(\mathbf{x}, w^*)$ of the "true dependency" in the class of approximating functions, from the available (finite) training data.
- Tell us *what* to do with the data.

Leaning Method

- Specifies *how* to obtain an estimate.
- A constructive implementation of an inductive principle for selecting an estimate $f(\mathbf{x}, w^*)$ from a particular set of functions $f(\mathbf{x}, w)$.
- For a given inductive principle, there are many learning methods corresponding to a different set of functions of a learning machine.

Common Learning Tasks

- Classification
- Regression
- Density estimation
- Clustering

Classification

- Output of learning machine need only take on two values.

- Loss function: classification error

$$L(y, f(\mathbf{x}, \omega)) = \begin{cases} 0 & \text{if } y = f(\mathbf{x}, \omega) \\ 1 & \text{if } y \neq f(\mathbf{x}, \omega) \end{cases}$$

- Risk functional: Quantifies the probability of misclassification
- Learning: To find the indicator function $f(\mathbf{x}, w_0)$ minimizing the probability of misclassification.

$$R(\omega) = \int L(y, f(\mathbf{x}, w)) p(\mathbf{x}, y) d\mathbf{x} dy$$

Regression

- Process of estimating a real-valued function based on a finite set of noisy samples.

$$L(y, f(\mathbf{x}, \omega)) = (y - f(\mathbf{x}, \omega))^2$$
$$R(\omega) = \int (y - f(\mathbf{x}, \omega))^2 p(\mathbf{x}, y) d\mathbf{x} dy$$

The risk can be decomposed into two terms:

$$\begin{aligned} R(\omega) &= \int (y - g(\mathbf{x}) + g(\mathbf{x}) - f(\mathbf{x}, \omega))^2 p(\mathbf{x}, y) d\mathbf{x} dy \\ &= \int (y - g(\mathbf{x}))^2 p(\mathbf{x}, y) d\mathbf{x} dy + \int (f(\mathbf{x}, \omega) - g(\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x} \\ &= \int E_\varepsilon(\varepsilon^2 \mid \mathbf{x}) p(\mathbf{x}) d\mathbf{x} + \int (f(\mathbf{x}, \omega) - g(\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x} \end{aligned}$$

Since the first term (noise variance) does not depend on w, minimizing the second term (function approximation error) is equivalent to minimizing $R(\omega)$
Thus, obtaining the smallest prediction risk is equivalent to the most accurate estimation of the unknown function $g(\mathbf{x})$ by a learning machine.

Density Estimation

- Output of the "system" is not used.
- The output of the "learning machine" represents density, i.e. $f(\mathbf{x}, \omega), \omega \in \Omega$ becomes a set of densities
- Loss function $L(f(\mathbf{x}, \omega)) = -\ln f(\mathbf{x}, \omega)$
- Risk functional $R(\omega) = \int -\ln f(\mathbf{x}, \omega) p(\mathbf{x}) d\mathbf{x}$
- Minimizing $R(\omega)$ using the training data $\mathbf{x}_1, \ldots, \mathbf{x}_n$ leads to the density estimation.

Clustering and Vector Quantization

- Risk functional: Optimal partitioning of the unknown distribution in $\mathbf{x}$-space into a prespecified number of regions.
- Future samples can be approximated by a single point (cluster center or local prototype).
- The set of vector-valued functions $f(\mathbf{x}, \omega), \omega \in \Omega$ are vector quantizers

$$\mathbf{x} \xrightarrow{f(\mathbf{x}, \omega)} c(\mathbf{x})$$

- Loss function: $L(f(\mathbf{x}, \omega)) = (\mathbf{x} - f(\mathbf{x}, \omega)) \cdot (\mathbf{x} - f(\mathbf{x}, \omega))$
- Risk functional: $R(\omega) = \int (\mathbf{x} - f(\mathbf{x}, \omega)) \cdot (\mathbf{x} - f(\mathbf{x}, \omega)) p(\mathbf{x}) d\mathbf{x}$

- **Vector quantizer** minimizing $R(\omega)$ optimally quantizes future data generated from generated from $P(\mathbf{x})$
- **Dimensionality reduction** finds low-dimensional mappings of a high-dimensional distribution.

Summary

- Loss functions

  Classification $$L(y, f(\mathbf{x},\omega)) = \begin{cases} 0 & \text{if } y = f(\mathbf{x},\omega) \\ 1 & \text{if } y \neq f(\mathbf{x},\omega) \end{cases}$$

  Regression $$L(y, f(\mathbf{x},\omega)) = (y - f(\mathbf{x},\omega))^2$$

  Density Estimation $$L(f(\mathbf{x},\omega)) = -\ln f(\mathbf{x},\omega)$$

  Clustering and Vector Quantization
  $$L(f(\mathbf{x},\omega)) = (\mathbf{x} - f(\mathbf{x},\omega)) \cdot (\mathbf{x} - f(\mathbf{x},\omega))$$

- Risk Functional

  Classification $$R(\omega) = \int L(y, f(\mathbf{x},w))\, p(\mathbf{x},y)\, d\mathbf{x}dy$$

  Regression $$R(\omega) = \int (y - f(\mathbf{x},\omega))^2\, p(\mathbf{x},y)\, d\mathbf{x}dy$$

  Density Estimation $$R(\omega) = \int -\ln f(\mathbf{x},\omega)\, p(\mathbf{x})\, dx$$

  Clustering and Vector Quantization
  $$R(\omega) = \int (\mathbf{x} - f(\mathbf{x},\omega)) \cdot (\mathbf{x} - f(\mathbf{x},\omega))\, p(\mathbf{x})\, d\mathbf{x}$$

Inductive Principles

- Maximum likelihood (ML)
- Empirical risk minimization (ERM)
- Penalized ERM

Maximum Likelihood (ML)

- Given a set of i.i.d. training data $\mathbf{X} = [\mathbf{x}_1,...,\mathbf{x}_n]$, the probability of $\mathbf{X}$ as a function of

t

$$P(\mathbf{X} \mid \omega) = \prod_{i=1}^{n} f(\mathbf{x}_i, \omega)$$

- We should choose the parameters ω which maximize the likelihood function
- This corresponds to choosing a ω^*, i.e., model $f(\mathbf{x}, \omega^*)$, which is most likely to generate $\mathbf{X}$
- To make problem more tractable *log likelihood function is used*. Thus, ML risk functional is

$$R_{\text{ML}}(\omega) = -\ln P(\mathbf{X} \mid \omega) = -\sum_{i=1}^{n} \ln f(\mathbf{x}_i, \omega)$$

Empirical Risk Minimization (ERM)

- Empirically estimates the risk functional using training data
- Empirical risk: average risk for the training data minimized by choosing the appropriate parameters
- Expected risk (density estimation)

$$R(\omega) = \int L(f(\mathbf{x},\omega))\, p(\mathbf{x})\, d\mathbf{x}$$

- This expectation is estimated by taking an average of the risk over the training data

$$R_{emp}(\omega) = \frac{1}{n} \sum_{i=1}^{n} L(f(\mathbf{x}_i, \omega))$$

- The optimum parameter values ω^* are found by minimizing the empirical risk with respect to ω.
- ERM is more general than ML since it does not specify the particular form of the loss function. ERM is equivalent to ML for density estimation, if the loss function is $L(f(\mathbf{x}, \omega)) = -\ln f(\mathbf{x}, \omega)$

Penalization (Regularization) Inductive Principle

- Assumes a flexible class of approximating functions $f(\mathbf{x}, \omega)$, $\omega \in \Omega$, where Ω is a set of abstract parameters.
- Penalization (regularization) term is used to restrict solutions

$$R_{pen}(\omega) = R_{emp}(\omega) + \lambda \Phi[f(\mathbf{x}, \omega)]$$

$R_{emp}(\omega)$ is a usual empirical risk

$\Phi[f(\mathbf{x}, \omega)]$ is a penalty (nonnegative, independent of data)

λ (> 0): regularization parameter

- A priori knowledge is included in penalty form.
- Regularization parameter λ

  Controls the strength of priori knowledge

  Very large λ → result of minimizing $R_{emp}(\omega)$ does not depend on data

  Small λ → final model does not depend on penalty functional

  Optimal value of λ is chosen using resampling methods

  Optimal model estimate is found as a result of a trade-off between fitting the data and a priori knowledge (i.e., a penalty term).

1.4 Learning as a Continuous Lifelong Process

Dimensions of classifying learning types

1. Generator of *input* data: passive vs. active learning
2. Types of *output* data: recognition vs. recall
3. Types of *teaching signals*: supervised vs. unsupervised vs. reinforcement learning
4. Nature of *interaction* with the environment (teacher): interactive vs. batch learning

Classical view of machine learning

- Passive rather than active
- Non-incremental rather than incremental
- Life-limited rather than life-long
- Exploitative rather than explorative
- Discriminative rather than generative
- Stationary rather than dynamic environment

Continuous learning: An alternative view

- **Definition**. A **continuous learning system** is a natural or artificial system that *continuously* improves its performance by *gaining information* from experiential data through *active* interaction with a *changing* environment.

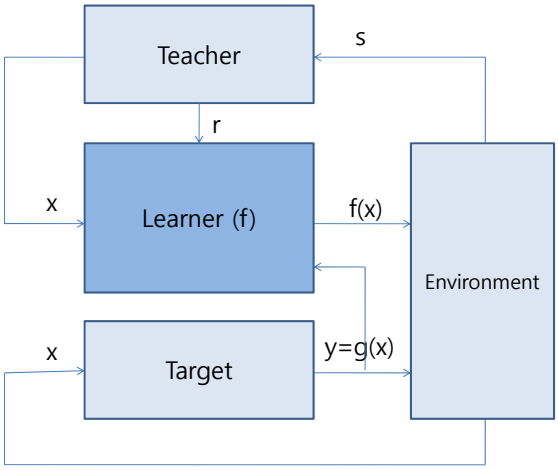

Figure 1-4. Continuous learning: Learner as a self-teaching cognitive agent in a dynamic environment. The learner (f) incorporates the teacher as its component in the continuous learning loop. The teacher observes the state (s) of the environment to generate the teaching signals of new inputs (x), rewards (r), and/or target outputs (y).

Example. Continuous learning in children's language acquisition

- Learners actively produces words and sentences (active)
- Current learning is influenced by previous learning (incremental, continuous)
- Language learning is a process over lifetime (life-long)
- Children do not just imitate the parents, but they generate new sentences (explorative)
- Children learns to recognize as well as recall linguistic expressions (generative).
- Language is changing and the contexts are varying (dynamic)

Objective function of the continuous learner:

Let $I(P, Q)$ be some measure of mutual information between probability distributions P and Q. Having a limited capacity of sample memory of size n, $D^t = x_{t:n} = (x_{t-n+1}, \dots, x_{t-2}, x_{t-1}, x_t)$, a continuous learner $f_t = f(x; x_{t:n})$ imitates the target system g by generating new examples and maximizing the mutual information $I\big(P_g(x), P_{f_t}(x)\big)$ between the target distribution $P_g(x)$ and the learner's distribution $P_{f_t}(x)$. The ultimate objective of the continuous learner is to maximize the lifelong, cumulative information gain

$$\lim_{t\to\infty} \sum_{i=1}^{t} I\big(P_g(x), P_{f_t}(x)\big)$$

where t is the time step of learning the most-recent data set of size n, $D^t = x_{t:n}$.

Towards self-teaching cognitive agents

- Continuous learning
- Psychological plausibility
- Biological basis of learning
- "Cognitive" machine learning

(End of Chapter 1)