Machine Learning

Chapter 6 Bayesian Learning

6.5 Maximum Likelihood Hypotheses For Predicting Probability

- Learn a nondeterministic function
  - Why?
    - Deterministic function f: $X \rightarrow \{0,1\}$
    - The unpredictability could arise from our inability to observe all data features
    - Need f is a probability function of the input
  - Task： learn the target function
    - f’: $X \rightarrow [0,1]$， such that f’=P(f(x)=1)

Two Ways

- Brute-Force way
  - collect the observed frequencies of 1's and 0's for each possible value of x
  - then train the neural network to output the target frequency for each x
- Training a ANN directly from the observed training examples of f, yet still derive a ML hypothesis for f'

Derive the criterion for optimization

– $D=\{<x_1,d_1>...<x_m,d_m>\}$

– $$P(D|h)=\prod_{i=1}^{m}P(x_i,d_i|h)=\prod_{i=1}^{m}P(d_i|h,x_i)P(x_i)$$

– Notice:

$$P(d_i|h,x_i)=\begin{cases}h(x_i) & d_i=1\\ 1-h(x_i) & d_i=0\end{cases}=h(x_i)^{d_i}(1-h(x_i))^{1-d_i}$$

$$P(D|h)=\prod_{i=1}^{m}h(x_i)^{d_i}(1-h(x_i))^{1-d_i}P(x_i)$$

$$h_{ML}=\arg\max_{h\in H}\prod_{i=1}^{m}h(x_i)^{d_i}(1-h(x_i))^{1-d_i}p(x_i)$$

$$=\arg\max_{h\in H}\prod_{i=1}^{m}h(x_i)^{d_i}(1-h(x_i))^{1-d_i}$$

$$=\arg\max_{h\in H}\sum_{i=1}^{m}d_i\ln h(x_i)+(1-d_i)\ln(1-h(x_i)) \qquad (6.13)$$

– Note the similarity between (6.13) and the general form of the entropy function in chapter 3.

$$\sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i)\ln(1 - h(x_i))$$

$$\sum_{i}^{c} - p_i \log_2 p_i$$

- the negation of the (6.13) is called the cross entropy

6.5.1 the gradient search to maximize likelihood in a neural net

- Obtain the maximum likelihood hypothesis(G(h,D)) for (6.13)
  - derive a weight-training rule for ANN learning

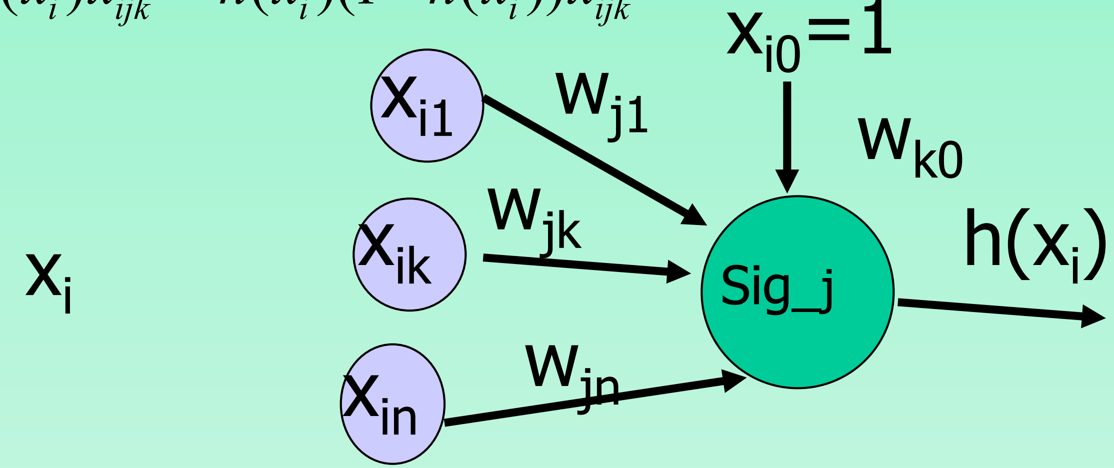
$$\frac{\partial G(h,D)}{\partial w_{jk}} = \sum_{i=1}^{m} \frac{\partial G(h,D)}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}}$$

$$= \sum_{i=1}^{m} \frac{\partial (d_i \ln h(x_i) + (1-d_i)\ln(1-h(x_i)))}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}}$$

$$= \sum_{i=1}^{m} \frac{d_i - h(x_i)}{h(x_i)(1-h(x_i))} \frac{\partial h(x_i)}{\partial w_{jk}}$$

– Suppose the ANN is constructed from a single layer of sigmoid units. We have

$$\frac{\partial h(x_i)}{\partial w_{jk}} = \sigma'(x_i)x_{ijk} = h(x_i)(1-h(x_i))x_{ijk}$$



– Maximize P(D|h), we perform gradient ascent

$$\frac{\partial G(h,D)}{\partial w_{jk}} = \sum_{i=1}^{m}(d_i - h(x_i))x_{ijk}$$

$$\Delta w_{jk} = \eta\sum_{i=1}^{m}(d_i - h(x_i))x_{ijk} \qquad w_{jk} \leftarrow w_{jk} + \Delta w_{jk}$$

- Compare with BP
  - Different in h_{ML}
    - Assumption for minimizing sum of squared error:
      - $<x_i,d_i>$， where $d_i=f(x_i)+e_i$
    - Assumption for minimizing cross entropy:
      - f': $X \rightarrow [0,1]$， such that $f'=P(f(x)=1)$

6.7 Bayes Optimal Classifier

- Two questions
  - “what is the most probable h given the training data?”
    - Learn a hypothesis
  - “what is the most probable classification of the new instance given the training data”
    - Classification

- Sample
  - Three hypotheses, h_1, h_2, h_3.
    - P(h1|D)=0.4, P(h2|D)=0.3, P(h3|D)=0.3
    - h_1 is the MAP hypothesis.
  - For a new instance x, h1(x)=1, h2(x)=0, h3(x)=0
  - Task: classification of x ?
    - MAP hypothesis: h1(x)=1
    - Taking all hypotheses into account, p(x=1)=0.4, and p(x=0)=0.6
    - We found the most probable classification is different from the classification given by h_{MAP}

- How to obtain the most probable classification
  - combining the predictions of all hypotheses, weighted by their posterior probabilities
  - $P(v_j|D)$ is the probability that the classification for the new instance is v_j .

$$P(v_j \mid D) = \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D)$$

  - The optimal classification is the value v_j for which $P(v_j|D)$ is maximum:

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D)$$

- example
  - given
    - possible classifications V={+,-}
    - $P(h_1|D)=0.4$, $P(-|h_1)=0$, $P(+|h_1)=1$
    - $P(h_2|D)=0.3$, $P(-|h_2)=1$, $P(+|h_2)=0$
    - $P(h_3|D)=0.3$, $P(-|h_3)=1$, $P(+|h_2)=0$
  - Therefor
    - $\sum_{h_i \in H} P(+|h_i)P(h_i|D) = 0.4$
    - $\sum_{h_i \in H} P(-|h_i)P(h_i|D) = 0.6$
    - $\arg\max_{v_j \in \{+,-\}, h_i \in H} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) = 0.6$

- Summary
  - Bayes optimal classifier maximizes the probability that the new instance is classified correctly
  - one curious property is that the predictions can correspond to a h not contained in H
    - Search of hypothesis space H'
      - perform comparisons between linear combinations of predictions from multiple hypotheses in H

6.8 Gibbs Algorithm

- Problem of the Bayes optimal classifier
  - obtains the best performance
  - quite costly to apply

- Gibbs algorithm, an alternative, less optimal method
  - Choose a h from H, according to the posterior probability distribution over H
  - Use h to predict the classification of the next instance x

- Classification error
  - at most twice the expected error of the Bayes optimal classifier

6.9 Naïve Bayes Classifier

- The learning tasks:
  - each instance x is described by a conjunction of attribute values,
  - the target function f(x) can take on any value from some finite set V,
  - predict the target value for new instance

- The learning method
  - Calculate the most probable target value v_{MAP}, given the instance data <$a_1,\ldots,a_n$>

$$v_{MAP} = \arg\max_{v_j} P(v_j \mid a_1,\ldots,a_n)$$

  - Use Bayes theorem to rewrite this expression

$$\begin{aligned} v_{MAP} &= \arg\max_{v_j \in V} \frac{P(a_1,\ldots,a_n \mid v_j)P(v_j)}{P(a_1,\ldots,a_n)} \\ &= \arg\max_{v_j \in V} P(a_1,\ldots,a_n \mid v_j)P(v_j) \end{aligned}$$

- Estimate the two terms in Bayes equation
  - $P(v_j)$: the frequency each target value v_j occurs in the training data
  - $P(a_1,...a_n|v_j)$: very difficult?
    - need a very, very large set of training data.
    - For simplify, we assume that the attributes values are conditionally independent given the target value

$$P(a_1,\ldots,a_n \mid v_j) = \prod_i P(a_i \mid v_j)$$

- Naïve Bayes classifier:

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$$

  - Notice: the number of $P(a_i|v_j)$, $P(a_1,\ldots,a_n|v_j)$
  - Whenever conditional independence is satisfied, this naïve Bayes classification v_{NB} is identical to the MAP classification
  - Difference from other learning methods?
    - there is no explicit search through the space of possible hypotheses

- An Example
  - Table 3.2 from chapter 3 provides a set of 14 training examples of the target concept PlayTennis, classify the following instance
    <sunny, cool, high, strong>
  - $$v_{NB} = \arg\max_{v_j \in \{yes, no\}} P(v_j) \prod_i P(a_i \mid v_j)$$
    $$= \arg\max_{v_j \in \{yes, no\}} P(v_j) P(sunny \mid v_j) P(cool \mid v_j) P(high \mid v_j) P(strong \mid v_j)$$
  - Calculate the probabilities of the different target values
    - P(yes)=9/14=0.64
    - P(no)=5/14=0.36
    - P(strong|yes)=3/9=0.33
    - P(strong|no)=3/5=0.60
    - ...
  - Calculate v_{NB}
    - P(yes)P(sunny|yes)P(cool|yes)P(high|yes)P(strong|yes)=0.0053
    - P(no)P(sunny|no)P(cool|no)P(high|no)P(strong|no)=0.0206
    - v_{NB}=no

- m-estimate
  - When the number of sample data is small, we use m-estimate $$\frac{n_c + mp}{n + m} \qquad (6.22)$$
    - p: prior estimate
    - m: a constant called the equivalent sample size
  - In the absence of other information, we assume uniform priors
    - if an attribute has k values, p=1/k
  - m is called the equivalent sample size
    - augmenting the n actual observations by an additional m virtual samples distributed according to p

6.10 An Example: Learning To Classify Text

- Learning problems:
  - Electronic news articles that I find interesting
  - Pages on web that discuss machine learning topics

- General settings for the naïve Bayes algorithm：
  - An instance space includes all possible text documents
  - training examples
  - target function f(x)
    - target value set V(V={like, dislike})

- The two main design issues in text classification
  - How to represent an arbitrary text document
  - How to estimate the probabilities

- Representing arbitrary text documents
  - define an attribute for each word position
  - define the value of that attribute to be the English word found in that position

- Document classification Sample
  - 700 documents that are classified as dislike
  - 300 that are classified as like
  - Task : classify the following new document.
    - document: This is an example document for the naive Bayes classifier. This document contains only one paragraph, or two sentences.
  - Calculate the naïve Bayes classification

$$v_{NB} = \underset{v_j \in \{like, dislike\}}{\arg\max} P(v_j) \prod_{i=1}^{19} P(a_i \mid v_j)$$

$$= \underset{v_j \in \{like, dislike\}}{\arg\max} P(v_j) P(a_1 = "this" \mid v_j) \ldots P(a_{19} = "sentences" \mid v_j)$$

– Note: In this problem, does the word probabilities for one text position are independent of the words that occurs in other positions hold?

- we have little choice but to make it
  - Without it, the number of probability terms is prohibitive

- In practice the naïve Bayes learner performs well in text classification problems

- Estimate $P(v_i)$ and $P(a_i=w_k|v_i)$.
  - $P(v_i)$: the fraction of each class in the training data
  - $P(a_i=w_k|v_i)$, do we need to estimate abut 2x50000x19 terms?
    - assume word is independent of the word position

$$P(a_i=w_k|v_i)=P(w_k|v_j)$$

    - The advantage： increase the number of examples available to estimate each of the required probabilities, thereby increasing the reliability of the estimates
  - Adopt m-estimate?

$$P(w_k \mid v_j) = \frac{n_k + 1}{n + |Vocabulary|}$$

- Learn_Naive_Bayes_Text(Examples, V)

  learn $P(w_k|v_j)$ and $P(v_j)$

  - Build Vocabulary
    - Vocabulary←collect all distinct words and tokens in Examples
  - Calculate $P(v_j)$ and $P(w_k|v_j)$
    - For each v_j in V do
      - $docs_j$←the subset of documents having the target value v_j
      - $P(v_j)$←$|docs_j|$ / |Examples|
      - $Text_j$←a single document created by concatenating all members of $docs_j$
      - n←total number of distinct word positions in $Text_j$
      - For each word w_k in Vocabulary
        - » n_k← number of times w_k occurs in $Text_j$
        - » $P(w_k|v_j)$←(n_k+1) / (n+|Vocabulary|)

- Classify_Naive_Bayes_Text(Doc)

  return the estimated target value for Doc.

  – a_i denotes the word at the position i in Doc
  – positions←All word position in Doc
  – Return v_{NB}， where

$$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i \mid v_j)$$

6.10 Experimental Results

- Joachims apply it to classify usenet news articles
  - The target classification for an article was the name of the usenet newsgroup
  - 20 electronic newsgroups, 1000 articles in each group
  - 3/2 of 20000 articles as training examples, the remaining third as test data.
  - The 100 most frequent words and any word occurring fewer than 3 times are removed in the Vocabulary
  - Accuracy: 89%
- Lang use it to learning the target concept “articles that I find interesting”
  - NewsWeeder
  - Three to four times as many interesting articles as the general articles