CS 6784: Spring 2010
Advanced Topics in Machine Learning Review

Guozhang Wang

February 25, 2010

1 Machine Learning Tasks

Machine learning tasks can be roughly categorized into three classes: supervised learning, unsupervised learning, reinforcement learning. Besides, there are some other learning tasks, such as semi-supervised learning, online learning, etc.

Supervised learning assume the data with feature X and label Y is i.i.d. sampled/generated from a distribution/process P(X, Y). The learner receive a portion of the data as training samples, can need to output the learning functions $h : X \rightarrow Y$ to predict the labels of test sample data.

Unsupervised learning also has the assumption of i.i.d. sampling from P(X). The data do not have a label, but only the observed features X. The learner needs to output somehow a "description" of the structure of P(X).

Reinforcement learning, however, does not hold the i.i.d. assumption. The input data is a markov decision process P(S — A, S') or P(R — S), along with a sequqnce of state/action/reward triples (s,a,r). The goal is to learn the "policy" that given a state S, generate the action A that maximizes the reward.

On the other hand, machine learning can also be treated as searching tasks in a hypotheses space, which is usually a very large space of possible hypotheses to fit 1) the observed data and 2) any prior knowledge held by the observer. Therefore, the common ways to do this is to narrow the space by settling on a parametric statistical model (space), and estimating parameter values by inspecting the data.

2 Supervised Learning

For supervised learning, the goal is to minimized a certain defined error. The *prediction error* (also called /generalization error/true error/expected loss/risk) is based on a hypothesis h for P(X, Y), and the *loss function*. The *sample error* is the tested error based on the test samples. When the sample size gets larger, the sample error approximates predication error better.

Now let's take the classification as an example to illustrate some other concepts in supervised learning. We assume training examples are generated by drawing instances at random from an unknown underlying distribution $P(X)$, then allowing a teacher to label this example with its Y value. From the Bayes's Decision Rule, we know that the optimal decision function is $argmax_{y \in Y}[P(Y = y|X = x)]$. Then the problem is how to get $P(Y = y|X = x)$ from the training data: one can definitely think of $P(Y = y|X = x) = \frac{P(X=x|Y=y)P(Y=y)}{P(X=x)}$. However, it is intractable to get full distribution of $P(X = x|Y = y)$, unless a tremendous number of samples provided **??**.

3 Generative vs. Discriminative Models

Given the complexity of learning $P(X = x|Y = y)$, we must look for ways to reduce this by making *independence assumptions*. This method is called the Naive Bayes Algorithm. In other words, we assume that feature attributes $X_1, X_2, ...X_n$ are all conditionally independent of one another given Y. Therefore:

$$argmax_{y \in Y}[P(Y = y|X = x)] = argmax_{y \in Y}[\frac{P(X=x|Y=y)P(Y=y)}{P(X=x)}]$$

Since the denominator $P(X = x)$ does not depend on Y:

$$\begin{aligned} &argmax_{y \in Y}[P(Y = y|X = x)] \\ &\quad = argmax_{y \in Y}[\tfrac{P(X=x|Y=y)P(Y=y)}{P(X=x)}] \\ &\quad = argmax_{y \in Y}[P(X = x|Y = y)P(Y = y)] \\ &\quad = argmax_{y \in Y}[P(X_1 = x_1, ..., X_n = x_n|Y = y)P(Y = y)] \end{aligned}$$

Therefore we can use maximum likelihood estimates or Bayesian MAP estimates to get the distribution parameter $\phi_{ij} = P(X = x_i|Y = y_j)$.

One should note that our original goal, $P(Y = y|X = x)$, has been transformed to $P(X = x|Y = y)P(Y = y) = P(X = x, Y = y)$ by the Bayesian Rule. This is actually an overkill to the original problem. This type of classifier is called a *generative* classifier, because we can view the distribution $P(X|Y)$ as describing how to generate random instances X conditioned on the target attribute Y with distribution $P(Y)$. Examples include naive Bayes, mixtures of multinomials, Bayesian networks, Markov random fields, and HMM.

Discriminative classifier, on the other hand, directly estimates the parameters of $P(Y|X)$. We can view the distribution $P(Y|X)$ as directly discriminating the value of the target value Y for any given instance X. For exmample, logistic regression first makes a parameterized assumption of the distribution $P(Y|X)$, and then tries to find the parameter ϕ to maximize $\Pi_{i=1}^{n}[P(Y = y_i)|X = x_i, \phi]$. This task can be done through MLE also. Some

other examples include neural networks, CRF, etc. Naive Bayes and Logistic Regression form a generative-discriminative pair for classification, as HMMs and linear-chain CRFs for sequantial data.

An even "direct" classifier would not even try to find the distribution $P(Y|X)$, but just a discriminant function that can predicate Y correctly from fewer training data. Examples include SVM, nearest neighbor, and decision trees.

As we introduce these three types of classifiers in turn, one can observe that the classifier's flexibility is decreasing, while the complexity is also decreasing, since we are targeting at smaller and smaller problems. Therefore when you have a lot of training data, the former ones might be a good choice, when you have few training data or the conditional distribution is supposed to be very complex, latter ones might be a good choice.

4 Hidden Markov Models

One key idea of general graphical models is to enforce conditional independence between variables and observation values through graphical structures. Hidden Markov Model is one that have strong assumptions of conditional independence between observations and states: one state is only depend on its direct predecessor (transition probability), and one observation is only depend on its corresponding state on the sequence (output/emission probability).

The learning of HMM is to estimate the transition and emission probabilities. Generative methods of maximum likelihood estimates have closed-form solutions.

The inference of HMM is to find the most likely state sequence. The problem is, the domain of possible state sequence is too large. Viterbi algorithm uses dynamic programming to solve this problem. It have runtime linear in length of sequence.

4.1 Graphical Models

Directed graphical models exploit conditional independence between random variables (i.e. states). HMM is one important example of directed graphical models. Undirected graphical models have more flexible representation of joint distribution. Important examples include Markov Networks and Markov Random Fields.

5 Support Vector Machines

Support Vector Machines (SVM) are learning systems that use a hypothesis space of *linear functions* in a high dimensional feature space, trained with a

learning algorithm from *optimization theory* that implements a learning bias derived from statistical learning theory. SVM is a discriminative method that brings together: 1) computational learning theory, 2) known methods in linear discriminant functions, and 3) optimization theory.

From the *computation learning theory* or *statistical learning theory*, we know that there is a trade-off between complexity and accuracy: if a model is too complex to be consistent with the training data, it would probably be *overfit*. One way to motivate this trade-off is to refer it to statistical bounds on the generalization error. These bounds will typically depend of certain quantities such as the *margin* of the classifier.

On the other hand, we are interested in techniques that will scale from toy problems to large realistic datasets. Thus we need to carefully choose the class of functions that is not only "rich" but also not very complex. One of the building blocks of SVM is the linear learning machines (usually apply perceptron learning algorithm), whose feature space can be implicitly defined in the *kernel functions*. Optimization theory gives SVM a precise characterization of these properties of the solution which guide the implementation of efficient learning algorithms.

We start introducing these three factors with the linear learning machines.

5.1 Linear Learning Machines

Nouikoff Theorem tells us that, the Rosenballt's Perceptron algorithm applied to linear classification converges in a finite number of iterations which is not depend on the scale of the data, provided its margin is positive. This gives the feasibility property of the margin: if it exists, then the algorithm will terminate in bounded time despite of the scale of training samples.

A second note is that, since the perceptron algorithm only updates the hyperplane if there is a misclassified data, and the the initial hyperplane is zero, then the final hyperplane is actually a linear combination of the training points. Therefore the linear machines (i.e. the algorithms used) has a dual representation of using just the data themselves. One important property of the duality is that data only appear through entries in the Gram matrix. In other words, in the decision function, *it is only the inner products of the data with the new point* that are needed. This property gives us an opportunity to increase the computational power of the linear learning machines by projecting the data into a high dimensional feature space implicitly, through *kernel functions.*

From the duality property we know that the decision function can be represented as a weighted sum of the inner products of points. If we have a way of computing the inner product in feature space directly as a function of the original input points, it is possible to merge the two steps needed to build a non-linear learning machine. Mercer's Theorem provides a characterization

of when a function is a kernel.

Another attraction of kernel function is that the learning algorithms and theory can largely be decoupled from the specifics of the application area, which must simply be encoded into the design of an appropriate kernel function.

5.2 Generalization Theory

Most importance of the role of margin, is that large margins suggests good generalization results (i.e., low generalization error). The proof lies in statistical learning theory, and here we only give a intuitive explanation **??**. Since the training and test data are assumed to have been generated by the same underlying dependence, we can consider that all the test points are generated by adding bounded pattern noise to the training patterns. If the noise bound is smaller than the margin, then we will correctly classify all test points.

More technically, from the Margin Error Bound Theorem, we know that the testing error bounds is mainly depend on the margin ρ given the fixed data radius bound. A large ρ leads to a small capacity term, but the margin error gets larger. A small ρ, on the other hand, will usually cause fewer points to have margins smaller than the margin, leading to a smaller margin error; but the capacity penalty will increase correspondingly. The overall message: *Try to find a hyperplane which is aligned such that even for a large ρ, there are few margin errors.*

We have thus accumulated a fair amount of evidence in favor of the following approach: Keep the margin training error small, and the margin large, in order to achieve high generalization ability. In other words, hyperplane decision functions should be constructed such that they maximize the margin, and at the same time separate the training data with as few exceptions as possible.

One reminder here is that, not all training data are linear separable. Thus we may not be able get a hyperplane with zero training error and non-zero margin. Thus a Soft-Margin Separation is needed, where the parameter C controls the trade-off between margin and training error. When C gets larger, we are intended to have less errors but with less margin, then the overfit effect would increase.

5.3 Optimization Theory

Maximizing ρ, however, is the same as minimizing the length of w. Hence we might just as well keep ρ fixed, say, equal to 1 (which is the case for canonical hyperplanes), and search for a hyperplane which has a small $\|w\|$ and few points with a margin smaller than $1/\|w\|$; in other words, few points such that $y* < w, x >< 1$.

Therefore, we can formalize the above training problem as a constraint optimization problem. Optimization theory tells us that this primal optimization problem can be transformed into its dual problem using the Lagrangian theory. The dual representation implies that only for input points that lies closet to the hyperplane are involved in deciding the hyperplane. Hence they are called the *support vector*. Thus this dual representation is stated to reduce computational complexity. The kernel trick allows us to map the data to arbitrarily high dimension without computation penalty.

As to the implementation, the dual problem can be solved using convex quadratic programming subject to linear constraints. Since convex quadratic programmes have no local maxima, their solution can always be found efficiently: gradient ascent is usually used for convex optimization problem.

5.4 Discriminant Function

Although linear discriminant function from $F(x, y : w) = w^T \Psi(x, y)$ has a limited expression power, it can already capture the generative methods for many graphical models. For example, the Hidden Markov Model tries to maximize the generalized model:

$$P(X = x|Y = y)P(Y = y)$$

$$= \Pi_{i=1}^{l} P(Y_c = y^{(i)}|Y_p = y^{(i-1)})P(X_c = x^{(i)}|Y_c = y^{(i)})$$

$$= \vec{w}^T \Phi(x, y)$$

with $w_{ab} = \log[P(Y_c = a|Y_p = b)]$ and $w_{cd} = \log[P(Y_c = c|Y_p = d)]$ and $\Phi(x, y)$ the feature histogram.

6 Conditional Random Fields

As we have talked before, discriminative models directly learn the conditional distribution over observations instead of the joint distribution over labels and observations. As a result, it needs not to enumerate all possible observations, and also can relax the conditional independence assumption often made by generative models so that it can leverage richer overlapping features of observations. On the other hand, there is another important factor of graphical models: directed models v.s. undirected models. Directed models have the biased label problem (especially for discriminative models), that the decision of the current label must be depend on the decision of the previous label (transition probability), and once the label is decided, it is fixed. Once the training data do not give the information for all possible transitions, it will make the wrong prediction.

Conditional Random Fields are discriminative and undirected models. It is evolved from Hidden Markov Model (generative and directed model), Markov Random Fields (generative and undirected), Maximum Entropy Markov Model (discriminative and directed). Its conditional distribution is based on the whole observations, therefore avoid the label bias problem. As a cost, its learning process needs to consider the normalization factor.

6.1 Model Comparison: HMM, SVM, M^{3}N, MEMM, CRF

The above models can be formalized in the same way, that the prediction process is to solve the following formulation:

$y = argmax_y[\sum \alpha^i_{y_{i-1},y_i}]$

For HMM, which is a generative model, the joint distribution is based on transition probability and emit probability: $\alpha^i_{y_{i-1},y_i} = log[P(y_i|y_{i-1}) * P(x_i|y_i)]$; For MEMM, which is a discriminative model, the conditional distribution is based on the transition probability involving the current label: $\alpha^i_{y_{i-1},y_i} = log[P(y_i|y_{i-1}, x_i)]$; For CRF, SVM, M^{3}N, AMN, they have the same linear feature function format: $\alpha^i_{y_{i-1},y_i} = w * \theta(y_{i-1}, y_i, x_i)$.

The learning process for HMM is to estimate transition probability and emit probability. It is "easy" (if there are not too many x_i) due to the conditional independence assumption. The learning process for MEMM is to estimate the transition probability involving the current label. It is actually a multi-class classification problem for every state.

The learning process for CRF, SVM, M^{3}N and AMN is to estimate the parameters for the linear feature functions. For CRF, it assume the conditional distribution has the form:

$\frac{exp(w*\theta(x,y))}{\sum_{\bar{y}} exp(w*\theta(x,\bar{y}))}$