Chapter 4

Logistic Regression as a Classifier

In this chapter, we discuss how to approximate the probability $P(y_q \mid S_p, x_q)$, i.e., the probability that if the underlying system is S_p, corresponding to a certain input x_q, the system's output is y_q. We explore a new memory-based method, *locally weighted logistic regression*, which aims at approximating $P(y_q \mid S_p, x_q)$ when the output y_q is categorical.

Figure 4-1 illustrates the task of this chapter. Suppose there is a system, S_p, whose input space is 2-dimensional, and the output is boolean. Suppose a unlabeled data point is (x_q, y_q), to approximate $P(y_q \mid S_p, x_q)$, we need some knowledge of system S_p. Memory-based methods assume that the knowledge comes from the previous observations of the system's behavior, i.e. the memory data points or the training data points, as the circles and crosses in Figure 4-1. The circles correspond to those memory data points of S_p with outputs equal to 0, the crosses correspond to the other memory data points with outputs equal to 1. Now, if there come two queries, residing at the positions of the dark triangles, if both of the queries' outputs are "cross", then intuitively $P((y_q = \text{"cross"}) \mid S_p, x_q = (2.0, 3.0)^T)$ should be close to 1.0 because the majority of its neighbors are crosses, while $P((y_q = \text{"cross"}) \mid S_p, x_q = (4.5, 1.0)^T)$ should be near 0.0, based on the similar reasoning.

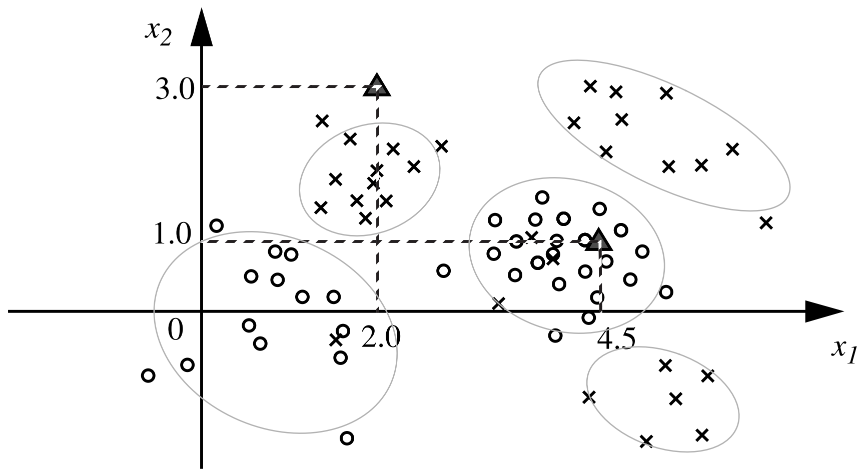


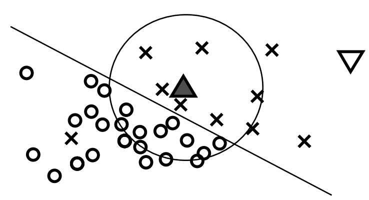
Figure 4-1: An illustration of the classification task.

4.1 Classification methods

Since the output y is categorical, the approximation of $P(y_q \mid S_p, x_q)$ is a classification problem by itself. System classification is to summarize a sequence of such classifications. There are many classification methods. The simplest one is nearest neighbor [Duda et al, 73; Aha et al, 89]. Its derivative, k-nearest neighbors, is more popular. Kernel regression, as mentioned in Chapter 2, is another important method. These methods are referred to as *memory-based* or *instance-based* classification methods [Atkeson et al, 97], while non-memory-based classification methods include neural network [Bishop, 95], decision-tree [Quinlan, 93], hierarchical mixtures of experts (HME) [Jordan, et al, 93], Bayes classifier [James, 85], etc. Both memory-based classifiers and non-memory-based ones assume the knowledge of the system S_p comes from the training data points. The distinguishing characteristic of memory-based classification methods is that they defer most of the processing of the training data points until after a query is made. This characteristic is desirable for processing continuous streams of training data and queries in real-time systems. In addition, the memory-based classifiers are capable of self-tuning according to the distribution and noise level of the training data points. Non-memory-based methods try to learn the underlying function model of the system S_p before any query comes. For example, neural networks have been proved capable of approximating any functions, if there is no restriction of the numbers of its hidden layers and its hidden nodes. Given a sufficient number of training data points, neural network uses them to approximate the underlying function relationship of the input and output. Once the training is done, the training data points are tossed away. Then, we wholly rely only on the trained neural network to process any queries.

Let's pick up some popular classification methods, and discuss them in a little depth.

1. Nearest neighborhood or *1*-nearest neighborhood doesn't perform satisfactorily in most cases, because it is too sensitive to the noise of the single nearest neighboring data point. *k*-nearest neighborhood performs quite well in many domains. But notice that it does not recognize the "boundary" of the different patterns. Besides, *k*-nearest neighborhood may be influenced by the density of the neighboring data points along the border. In the following diagram, intuitively the output of the query (the dark triangle) should be a cross,



because it is on the cross side. However, *k*-nearest neighborhood's conclusion tends to be a circle, because among the *k* nearest neighboring data points, the majority are circles.

2. Kernel regression is a good method for interpolation. However, it is not ideal for extrapolation. Suppose a query resides at a location remote from the centroid of other memory data points, like the reversed triangles in the above diagram, Kernel regression can not clearly decide if the category of the reversed triangle. Instead, it tends to assign 50% to the probability for the query's output to be "cross" (or "circle").

3. The simple Bayes classifier, referring to Section 3.3, puts too strong assumptions on the distribution of the data points. The conventional Bayes classifier assumes that if the out-

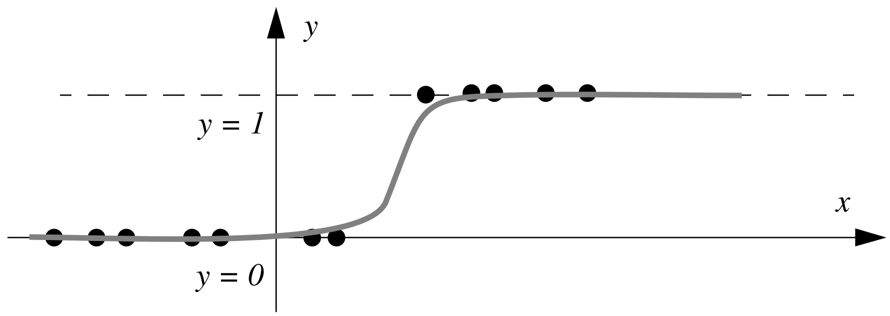
puts are boolean, the memory data points distribute in two clusters, one for the memory data points with output equal to 0, the other cluster for the data points with output equal to 1. Furthermore, the points are Gaussian-distributed in the input space so that the shapes of the clusters are ellipses. Referring to Figure 4-1, these restrictions are too strong for most classification problems. Even if we extend Bayes classifier to consider multiple clusters, it is still too hard to meet the requirement that the shapes of these clusters must be ellipses. Another concern about Bayes classifier is that it needs a large number of parameters to decide the centroids and the shapes of the Gaussian ellipses, this problem becomes more severe when we employ multiple ellipses.

4. The idea of a decision tree [Quinlan, 93] is to partition the input space into small segments, and label these small segments with one of the various output categories. However, conventional decision tree only does the partitioning to the coordinate axes. It is plausible that with the growth of the tree, the input space can be partitioned into tiny segments so as to recognize subtle patterns. However, overgrown trees lead to overfitting. More flexible than the conventional decision tree, CART [Breiman et al, 84] and Linear Machine Decision Tree [Utgoff et al, 91] can divide the input space using oblique lines. However, any nonlinear boundary may either make the tree overgrown or reduce the accuracy of the classification.

In this thesis, we explore a locally weighted version of logistic regression which can be used as a new memory-based classification method. Our method shares the properties of other memory-based classification methods. Besides, our method has some other good properties, including simplicity, capability of extrapolating, and a known confidence interval. Concerning the accuracy, our new method is competitive with others, supported by the experimental results.

4.2 Global logistic regression

Locally weighted logistic regression can be used to approximate $P(y_q \mid S_p, x_q)$. Let's begin with a very simple case with boolean output, shown in the following figure.



The straightforward way to approximate this function is to use two line segments to fit the dots, which are also referred to as training data points. However, to be learnable, we want to use a differentiable function to do the fitting instead of using two line segments. *Logistic* function, which is also referred to as *sigmoid* function, can be employed here. Logistic function is a monotonic, continuous function between *0* and *1*, whose shape is shown as the grey curve in the above figure. Mathematically, it is defined as,

$$\pi_q = \frac{1}{1 + \exp(-(1, \underset{\sim}{x}_q^T)\underset{\sim}{\beta})} \tag{4-1}$$

where $\underset{\sim}{x}_q$ is the input vector of the query, and $\underset{\sim}{\beta}$ is the parameter vector. π_q as the probability for y_q to be *1*, *i.e.*

$$\pi_q \equiv P(y_q = 1 | S_p, x_q) \qquad \text{Or, equivalently,}$$

$$y_q = \begin{cases} 1 & \text{with probability } \pi_q \\ 0 & \text{with probability } 1 - \pi_q \end{cases}.$$

Therefore, deciding the output of a query is now equivalent to finding the value of $\underset{\sim}{\beta}$.*Global* logistic regression assumes that all data points share the same parameter vector with the query, *i.e.*

$$\beta_1 = \beta_2 = \ldots = \beta_N = \beta$$

While *local* logistic regression allows $\underset{\sim}{\beta}_i$ vary cross the input space, but it changes smoothly. For example, if $\underset{\sim}{x}_1$ and $\underset{\sim}{x}_2$ are neighboring to each other, then we assume $\underset{\sim}{\beta}_1$ and $\underset{\sim}{\beta}_2$ must be close to each other, too. Back to global logistic regression, a good estimate of $\underset{\sim}{\beta}$ should fit, or in plain words, "go through", all the training data points as well as possible. Mathematically, the estimate of $\underset{\sim}{\beta}$ can be derived by maximizing the likelihood as following,

$$\begin{aligned} \mathrm{Lik}_G &= \prod_{i=1}^{N} P(y_i | S_p, x_i) = \prod_{i=1}^{N} \pi_i^{y_i}(1-\pi_i)^{1-y_i} \\ &= \prod_{i=1}^{N} \left[\frac{1}{1+\exp(-(1, \underset{\sim}{x}_i^T)\underset{\sim}{\beta})}\right]^{y_i} \left[\frac{\exp(-(1, \underset{\sim}{x}_i^T)\underset{\sim}{\beta})}{1+\exp(-(1, \underset{\sim}{x}_i^T)\underset{\sim}{\beta})}\right]^{1-y_i} \end{aligned} \tag{4-2}$$

Global logistic regression is a well-established algorithm in statistical literature [McCullagh et al, 89]. Although we discuss only the binary output case here, global logistic regression is ready to be extended to multiple categorical output cases. We will talk about this later.

The simplest classification problem is illustrated as Figure 4-2. The input is one-dimensional, which is represented by the horizontal axis; the output is boolean, represented by *0* or *1* on the vertical axis. The small circles in the pictures are the data points in memory. Global logistic regression works perfectly in the noise free case illustrated by Figure 4-2 (a), because the logistic function curve goes through most of the data points in memory. Global logistic regression also works in the noisy case shown as Figure 4-2 (b). Although the function curve moves midway between the data points, the curve is close to most of the data points. In summary, global logistic regression can be used as a noise tolerant classification method.

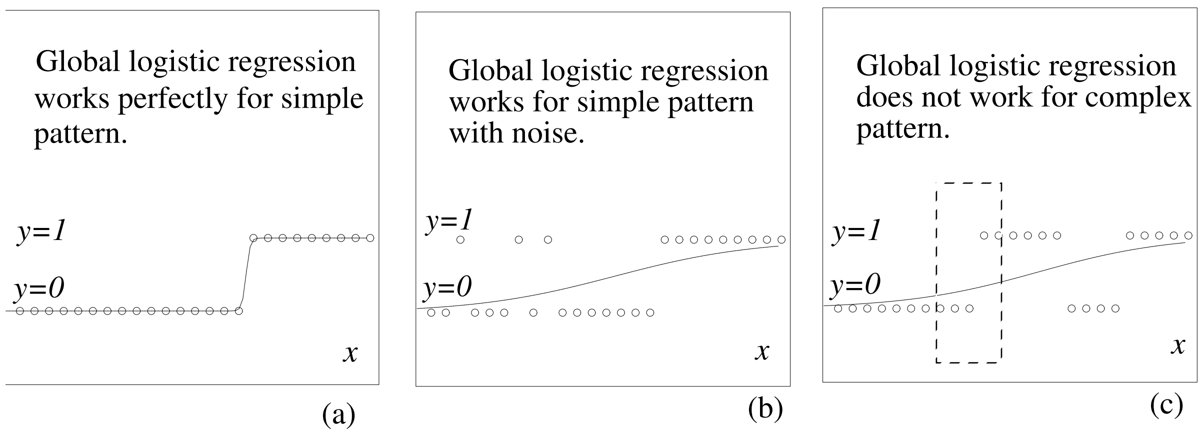


Figure 4-2: (Global) logistic regression for classification.

The fatal weakness of global logistic regression is shown in Figure 4-2 (c). Since it contains more than two segments, global logistic regression does not work. Recalling logistic function is a monotonic function, that is the reason global logistic regression fails whenever there are more than two segments. There are two approaches to solve this problem. One way to think is that although one logistic function does not work, we can combine several logistic functions. In fact, neural networks, especially feed-forward multi-layer perceptrons, can be regarded as an implementation of this idea.

The second approach resorts to the localization paradigm. The idea of local logistic regression is that although no single logistic function works well globally, in any local region a single function should be capable of doing the classification.

There are several versions of local logistic regression that can be investigated. *K*-nearest neighbor local regression would only select those neighboring data points, and ignores all others. Locally weighted version of logistic regression does not ignore any data points in memory, instead, it discriminates the data points by assigning weights to them.

4.3 Locally Weighted Logistic Regression

4.3.1 Maximum Likelihood Estimation

Locally weighted logistic regression is very similar to the global logistic regression, except that the locally weighted version assign a weight to $P(y_i|S_p, x_i)$. Differing from Equation 4-2, the locally weighted version of likelihood is,

$$\mathrm{Lik}_L = \prod_{i=1}^{N} P(y_i|S_p, x_i)^{w_i} \qquad \text{in which } w_i = \exp\left(-\frac{\mathrm{distance}(x_i, x_q)^2}{K_w^{\ 2}}\right) \tag{4-3}$$

The weight is a function of the Euclidean distance from the *i*'th memory data point to the query. Other metrics of distance are also possible depending on the specific domains. The K_w in the weighting function definition is referred to as *Kernel width*. The influence of Kernel width will be discussed shortly. Due to the weights, those data points remote from the query have smaller weights, while the neighboring memory data points have bigger weights.

Using Newton-Raphson algorithm, and through some algebraic manipulations, the maximum likelihood estimate of $\underset{\sim}{\beta}$ can be simplified as,

$$\hat{\beta}_{(r+1)} = \hat{\beta}_{(r)} + (X^T WX)^{-1} X^T We \tag{4-4}$$

Suppose there are N data points in the memory, each data point consists of a d-dimensional input vector and a boolean output. X is then a $N \times (1 + d)$ matrix. The *i*'th row of X matrix is $(1, x_i^{\mathrm{T}})$. And W is a $N \times N$ diagonal matrix, whose *i*'th diagonals element is, $W_i = w_i \pi'_i$, where π'_i is the derivative of π_i with respect to $\underset{\sim}{\beta}$, i.e.,

$$\pi_i' \equiv \frac{\exp(-(1, x_i^T)\beta)}{(1 + \exp(-(1, x_i^T)\beta))^2} \begin{pmatrix} 1 \\ x_i \end{pmatrix}. \tag{4-5}$$

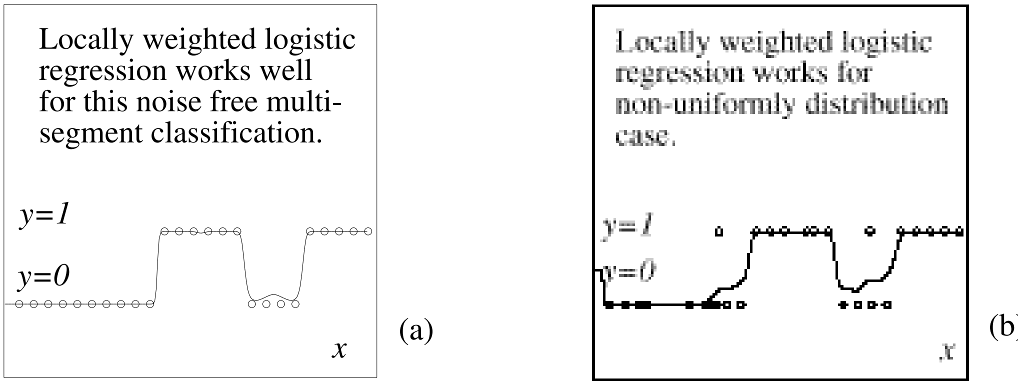


Figure 4-3: Locally weighted logistic regression as a classification technique works robustly.

w_i is the weight defined in Equation 4-3. The last item, e is a ratio of $y_i - \pi_i$ to π'_i. Newton-Raphson algorithm starts from a random vector of $\underset{\sim}{\beta}$, usually we assign $\hat{\underset{\sim}{\beta}}_{(0)}$ to be zero vector. The recursive process converges very quickly, usually no more than 10 loops. Once we get the maximum likelihood estimate of $\underset{\sim}{\beta}$, we can estimate the query's π_q.

Also notice that, $(X^tWX)^{-1}$ is the asymptotic variance matrix of $\hat{\underset{\sim}{\beta}}$.

Now let us go back to the case of Figure 4-2 (c) and see if locally weighted logistic regression classifier is capable of solving the problem where global logic regression fails. The result is shown in Figure 4-3 (a). The circles are the memory data points. And each dot on the solid curve, which is π_q, is plotted by doing its own locally weighted regression at that local region. Locally weighted logistic regression works well in this case. Also, in the harder case of Figure 4-3 (b), it still works. Notice, π_q is influenced by the noise but not the distribution of the data points.

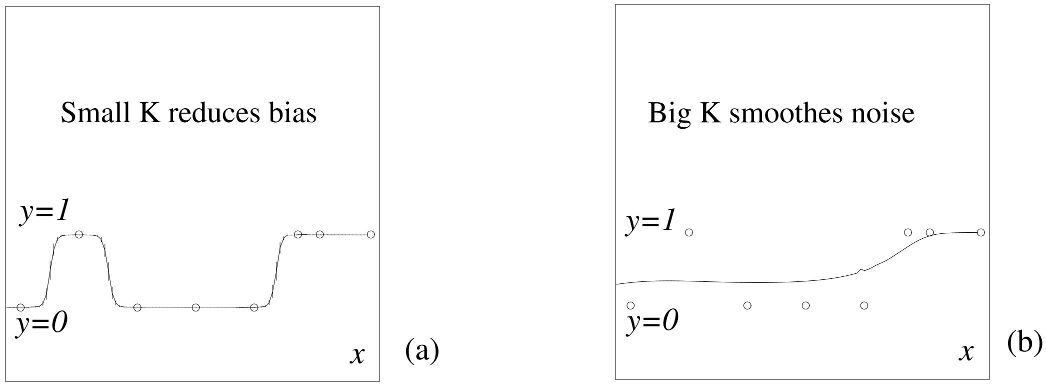


Figure 4-4: Kernel width adjusts the weighting function.

4.3.2 Weighting Function and Kernel Width

Referring to Equation 4-3, w_i, the weight can be adjusted by the Kernel width.When the Kernel width is big, more data points have high weights. Therefore, a big K_w is usually preferred when the noise level in memory is high. Extremely, when K_w goes to infinity, locally weighted logistic regression is equivalent to the global one. When K_w is small, only those close neighbors can effect the regression. Hence, a small K_w is good at recognizing the details of the memory. The influence of K_w is demonstrated by Figure 4-4.

4.3.3 Confidence Interval

Our estimate $\hat{\pi}_q$ is a point estimate, which is our best guess for the true value of π_q. Reporting only the point estimate is often unsatisfactory. Some measure of how close the point estimate is likely to be the true value is required. The *confidence interval* is such a metric.

The confidence interval of π_q is an interval of plausible values for π_q, $[\pi_L, \pi_U]$; the probability or the confidence for the true value of π_q falling into this interval is $100(1-\alpha)\%$, in which α is the *confidence level*. Usually we pre-define a confidence level, then decide the lower and upper bounds, π_L and π_U, which are also effected by the density and consistency (noise level) of the data points in the neighborhood of x_q.

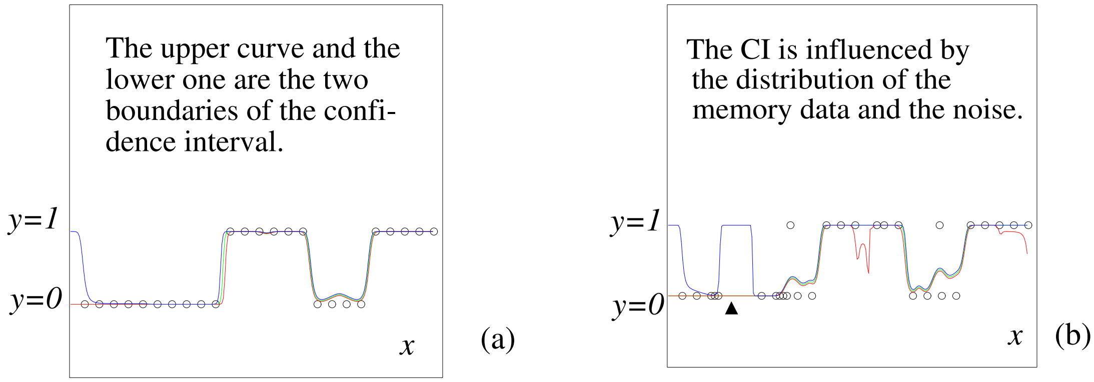


Figure 4-5: Confidence intervals for classification.

Referring to Equation 4-1, π is a monotonic function of $(1, x_q^T)\underset{\sim}{\beta}$; hence, to calculate the lower and upper bounds, we need know the lower and upper bounds of $(1, x_q^T)\underset{\sim}{\beta}$. Referring to Subsection 4.3.1, we can calculate the asymptotic variance of $\hat{\beta}_q$ which is $(X^T WX)^{-1}$, where X is decided by the memory data points and W is effected by the distances from the query to the memory data points. Notice that the asymptotic variance of $\hat{\beta}_q$ is likely to be small when there are more data points in the memory, especially in the neighborhood of the query. It is straightforward to calculate the confidence interval of $\hat{\pi}_q$ based on the upper and lower bounds of $x_q\hat{\beta}_q$.

The confidence intervals of the cases of Figure 4-3 (a) and (b) are plotted in Figure 4-5 (a) and (b). When the data points distribute uniformly as Figure 4-5 (a), the confidence interval is quite consistent. Otherwise, the confidence interval varies cross the input space.

According to π_q'’s definition, referring to Equation 4-5, when $\hat{\pi}_q$ is close to *1.0*, we tend to predict that the query’s output y_q is likely to be *1*. However, if at the same time $\hat{\pi}_q$’s confidence interval is too big, we should be conservative about our prediction. Figure 4-5 (b) shows such a situation: $\hat{\pi}_q$ is almost zero, therefore, if we only rely on $\hat{\pi}_q$, we should predict the output will be *0*. But since the confidence interval is very wide, we should be aware that there is still a lot of chance for the output y_q to be *1*.

Confidence interval is helpful for active learning and/or experimental designs. Wherever the confidence interval is wide, we need more data points in that region.

4.3.4 Multi-categorical classification inference

Up to now, we focus on boolean classification. In case the output has more than two output categories, locally weighted logistic regression method is still useful. But we should do some modifications.

1. Suppose there are m output categories, we can represent the output by a m-dimensional vector. If a data point falls into the first category, its output, $\underset{\sim}{y}$, is $[1, 0, \ldots, 0]^T$; if it is in the second category, $\underset{\sim}{y}$ is $[0, 1, \ldots, 0]^T$. In general, the distribution of output is multinomial, in the form of,

$$P(\underset{\sim}{y}_q | S_p, \underset{\sim}{x}_q) = \pi_1^{\underset{\sim}{y}_q} \pi_2^{\underset{\sim}{y}_q} \ldots \pi_m^{\underset{\sim}{y}_q} = \prod_{j=1}^{m} \pi_j^{\underset{\sim}{y}_q}$$

where π_j the probability for the data point falling into the j'th category.

2. We assume π_j is decided by a function similar to logistic function,

$$\tau_j = (\exp((1, \underset{\sim}{x}_q^T)\underset{\sim}{\beta}_j)) / \sum_{j=1}^{m} \exp((1, \underset{\sim}{x}_q^T)\underset{\sim}{\beta}_j)$$

Notice that the sum of π_j, $j = 1, \ldots, m$, is *1.0*. And for each output category, there is a unifying $\underset{\sim}{\beta}_j$; totally, there are m of them.

3. The likelihood can be constructed following the descriptions in Section 4.2 and section 4.3. For example, the global likelihood, which assumes all data points share the same $\underset{\sim}{\beta}$, is defined by Equation 4-6,

$$\mathrm{Lik}_G(\underset{\sim}{\beta}) = \prod_{i=1}^{N} P(\underset{\sim}{y}_i | S_p, \underset{\sim}{x}_i) = \prod_{i=1}^{N} \prod_{j=1}^{m} \pi_j^{y_i} \tag{4-6}$$

Now it is straightforward to follow the same inferences described in Sections 4.3 to figure out the locally weighted regression of $\underset{\sim}{\beta}_q$ and the confidence interval of π_q.

4.4 Comparison Experiment

Artificial Experiments

We artificially generate three data sets, each data consists of two input attributes (*2-d* input) and a boolean output. In Figure 4-6, we represent those data points with output values equal to *0* by circles, and represent the other data points, whose outputs are *1*, by crosses.

Figure 4-6 (a-c) are the contours of the $\hat{\pi}_q$ values corresponding to three different memory data sets. Figure 4-6 (a) shows a simple case, in which locally weighted logistic regression does a perfect job. Figure 4-6 (b) is similar to Figure 4-6 (a) except that, the "boundary" of the two regions is messier, and there is noise involved as well. In this case, $\hat{\pi}_q$ value increases from *0* to *1*, starting from the bottom left corner to the top right one; hence, locally weighted logistic regression works well, too. The small gradient of the contour of $\hat{\pi}_q$ shows the influence of the inconsistency (noise) of the data points in memory. Figure 4-6 (c) is the hardest case, in which locally weighted logistic regression still works well. Figure 4-6 (d) is the contour of *confidence interval* for the same memory as Figure 4-6 (c). It is apparent that the memory data points' noise level, as well as their distribution and density, influence the confidence interval.

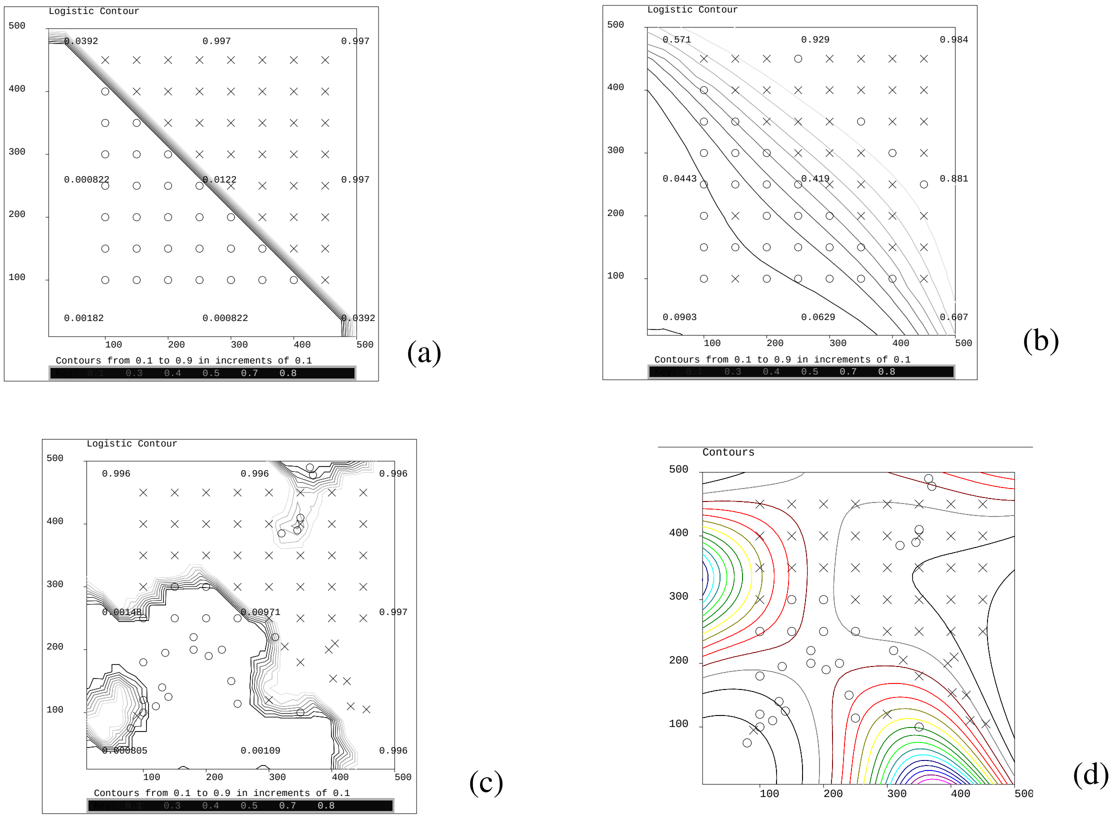


Figure 4-6: Three artificially generated data sets as the testbeds of locally weighted logistic regression classifier.

Real World Datasets

We use four binary output data sets from UCI's machine learning dataset repository, Ionos., Pima., Breast., and Bupa. We try six different classifiers, including nearest neighbor method (*1-Nearest*), *k*-nearest neighbors (*k-Nearest*), Kernel regression (Kernel), conventional Bayes classifier with two clusters (Bayes), C4.5 decision tree (*Decision*), feedforward perceptron (*Neural*), global logistic regression (*Global Logistic*) and our locally weighted logistic regression method (*Local Logistic*). The dimensionalities of the inputs vary from *6-d* to *34-d*.

We split each data set into two parts, the first part contains two thirds of the data points, which are used as the memory or the training dataset. The remaining one third of the data points are used as the test set. We can approximate the accuracy of a certain method for a certain dataset by the *error rate*, which is the ratio of the number of the failures to the number of the testing data points. For the same dataset, the lower the error rate, the better the classification method performs.

With different Kernel width, locally weighted logistic regression may have different accuracies. We split the range of the Kernel width into ten equal-length steps, and tried the logistic regressions using these ten different Kernel widths, so as to find the optimal Kernel width. Similarly, for k-nearest neighbor method and kernel regression, we enumerated parameter k from *10* to *100* with step *10*; for perceptron, we tried one-hidden layer feedforward perceptron with *1* to *10* hidden nodes. In this way, we found the best parameters for the various machine learning methods.

For each dataset, we shuffled it five times; each time we split it into training set and testing set. Hence, for each dataset by each method, we got five error-rates which were the best performances of the method with the tuned-up parameter(s). We recorded the mean values of these error-rates in Table 4-1, along with the standard deviations in parentheses.

Table 4-1: Comparison of logistic classifier with other methods

Error rate (%)	Ionos. (34-d)	Pima (8-d)	Breast (9-d)	Bupa (6-d)
1-Nearest	12.7 (2.5)	33.9 (1.8)	4.9 (0.6)	40.0 (2.4)
k-Nearnest	13.9 (2.9)	31.5 (4.7)	3.3 (0.5)	37.5 (5.8)
Kernel	12.7 (3.3)	30.9 (3.2)	3.3 (0.6)	37.3 (2.0)
Bayes	12.9 (1.2)	25.3 (2.3)	3.4 (1.2)	34.2 (3.6)
Decision	9.2 (2.1)	28.6 (3.0)	4.2 (1.1)	35.8 (3.2)
Neural	10.5 (3.2)	33.4 (2.0)	3.2 (0.6)	32.1 (4.5)
Global Logistic	12.4 (0.7)	24.9 (3.0)	3.9 (1.4)	34.4 (3.4)
Local Logistic	13.0 (0.4)	22.5 (2.8)	3.1 (0.7)	31.0 (2.7)

The experiments show that the accuracy of the locally weighted logistic regression method (Local Logistic) is competitive compared with other classification method. Some remarks are listed as following,

1. It is not surprising that locally weighted logistic regression is more accurate in most cases than *1*-nearest neighborhood, *k*-nearest neighborhood, Kernel regression, conventional Bayes classifier, C4.5 decision tree, and global logistic regression according to our discussion in Section 4.1.

2. Global logistic regression's performance is similar to that of the conventional Bayes classifier with two clusters. But global logistic regression is computationally cheaper than the conventional Bayes classifier. Suppose the input space's dimensionality is d and the memory size is N, the computation cost of locally weighted logistic regression is $O(d^3 + d \times N)$, while that of the conventional Bayes classifier with improved efficiency by some tricks is $O(d^3 \times N + d \times N \times k)$, where k is the number of clusters.

3. Concerning neural networks, locally weighted logistic regression does not outperform it in accuracy. Instead, an advantage comes from the general good properties of the memory-based approach over non-memory-based ones. As mentioned in the beginning of this chapter, Section 4.1, as well as [Atkeson et al., 97], because memory-based learning does not process data until the query arrives, the parameters of the logistic regression are not fixed in advance. When we update the memory, unlike neural network, less interference will happen, because the previous arrived memory data points are treated equally as the new comers. And by adjusting the parameters, we can shift the logistic regression continuously along the global-local spectrum.

4. Locally weighted logistic regression performs poorly on the Ionos dataset. The reason is that the dimensionality of the input is very high (34-d). Maybe many input attributes are irrelevant to the classification but only confuse the classifiers. When we selected the first,

the fourth and the fifth attributes to be input, the mean value of error-rate of the local logistic classifier dropped from *13.0%* to *10.7%*, with standard deviation *0.7%*.

To eliminate those less important input variables, recall that locally weighted logistic regression estimates the parameter vector β. In fact, each element of β indicates the significance of the corresponding input attribute for classification. If one element of β is close to zero, it implies that the corresponding input attribute is not very relevant to the classification job. We can get rid of the irrelevant input attributes using this heuristic. Some preliminary experiments showed that the selection result was quite consistent with the nodes of decision tree.

4.5 Summary

In this thesis, we explore a locally weighted version of logistic regression which can be used as a new memory-based classification method. Our method shares the properties of other memory-based classification methods. Besides, our method has some other desirable properties, including simplicity, competitive accuracy, capability of extrapolating, and confidence interval.

In Chapter 5 and Chapter 6, we will discuss the issue about how to improve the efficiency of locally weighted logistic regression as well as other memory-based methods.