Incremental Weighted Naive Bayes Classifiers for Data Stream

Christophe Salperwyck[1], Vincent Lemaire[2], and Carine Hue[2]

[1] Powerspace, 13 rue Turbigo, 75002 Paris
[2] Orange Labs, 2 avenue Pierre Marzin, 22300 Lannion

Abstract. A naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem with naive independence assumption. The explanatory variables (X_i) are assumed to be independent from the target variable (Y). Despite this strong assumption this classifier has proved to be very effective on many real applications and is often used on data stream for supervised classification. The naive Bayes classifier simply relies on the estimation of the univariate conditional probabilities $P(X_i|C)$. This estimation can be provided on a data stream using a "supervised quantiles summary". The literature shows that the naive Bayes classifier can be improved (i) using a variable selection method (ii) weighting the explanatory variables. Most of these methods are related to batch (off-line) learning and need to store all the data in memory and/or require reading more than once each example. Therefore they cannot be used on data stream. This paper presents a new method based on a graphical model which computes the weights on the input variables using a stochastic estimation. The method is incremental and produces an Weighted Naive Bayes Classifier for data stream. This method will be compared to classical naive Bayes classifier on the Large Scale Learning challenge datasets.

1 Introduction

Since the 2000s, the data mining from streams became a standalone research topic. Many studies addressing this new problem have been proposed (Gama (2010)). Among the solutions to the problems of learning on data streams, the incremental learning algorithms are one of the most used techniques. These algorithms are able to update their model using just the new examples.

In this article we focus on one of the most used classifier in the literature: the naive Bayes classifier. We modify this classifier in order to make on-line supervised classification on data streams. This classifier only needs conditional probability $P(X_i|C)$, with X_i an explanatory variable and C a class of the classification problem. Its complexity to predict is very low which makes it suitable and widely used for stream mining prediction.

Nevertheless it has been proved for batch learning that selecting (Koller and Sahami (1996)), Langley and Sage (1994)) or weighting (Hoeting et al. (1999)) variables can improve the classification results. Moreover Boullé in (Boullé (2006b)) shows the close relation between weighting variables and averaging many naive Bayes classifier in the sense that at the end the two different processes produce a similar single model (see equation 1). In this paper we particularly focus on weighting variables for data streams for a naive Bayes classifier. This weighting produces a single model close to an "averaged naive Bayes classifier" a "weighted naive Bayes classifier". We propose a graphical model and a learning method to compute these weights.

The present work aims to study a new way to estimate incrementally the weights of a Weighted Naive Bayes classifier (WNB) using a graphical model close to a neural network. The paper is organized as follow: our graphical model and the way to compute its parameters (the weights) are presented in section 2 ; the section 3 presents how the conditional density estimations, used as inputs of our model, are estimated ; the section 4 realizes an experimental study of our Weighted Naive Bayes classifier trained incrementally on the large scale learning challenge datasets. Finally the last section concludes this paper.

2 Incremental Weighted Naive Bayes classifiers

2.1 Introduction: Naive Bayes Classifier (NB) and Weighting Naive Bayes Classifiers (WNB)

The naive Bayes classifier (Langley et al. (1992)) assumes that all the explanatory variables are independent knowing the target class. This assumption drastically reduces the necessary computations. Using the Bayes theorem, the expression to obtain the estimation of the conditional probability of a class C_k is: $P(C_k|X) = \frac{P(C_k)\prod_i P(X_i|C_k)}{\sum_{j=1}^{K}(P(C_j)\prod_i P(X_i|C_j))}$ where K is the number of classes, i the index of the explanatory variable.

The predicted class is the one which maximizes the conditional probabilities $P(C_k|X)$. The probabilities $P(X_i|C_k)$ are estimated using a conditional probability density estimation as for example using counts after discretization for numerical variables or grouping for categorical variables. The denominator of the equation **??** normalizes the result so that $\sum_k P(C_k|X) = 1$. One of the advantages of this classifier in the context of data stream is its low complexity for deployment, which only depends on the number of explanatory variables. Its memory consumption is also low since it requires only one conditional probability density estimation per variable.

The literature shows that the naive Bayes classifier could be improved (i) using a variable selection method (Koller (1996), Langley and Sage (1994)) (ii) weighting the explanatory variables which amounts to a Bayesian Model

Averaging (BMA) (Hoeting et al. (1999)). These two processes can be mixed iteratively. The formulation of the conditional probabilities becomes:

$$P(C_k|X) = \frac{P(C_k)\prod_i P(X_i|C_k)^{w_i}}{\sum_{j=1}^{K}(P(C_j)\prod_i P(X_i|C_j)^{w_i})} \quad (1)$$

where each explanatory variable i is weighted by a weight w_i ($w_i \in [0-1]$).

2.2 The proposed approach

In off-line learning, the weights of the Weighted Naive Bayes Classifiers can be estimated using (i) an averaging of the Naive Bayes classifiers obtained (Hoeting et al. (1999)) (ii) an averaging of the Naive Bayes classifiers obtained using a MDL (Minimum Description Length) criterion (Boullé (2006b)) (iii) a direct estimation of the weights using a gradient descent (Guigoures and Boullé (2011)). But all these methods require to have all the data in memory and to read them several times. The method proposed in this paper optimizes directly the weights of the classifier and is able to work on data stream.

The first step has been to elaborate a graphical model (see Figure 1) dedicated to the optimization of the weights. This model allows the rewriting of the equation 1 as a graphical model where the Weighted Naive Bayes classifier has a weight per class and per variable as presented in the equation 2. The number of weights is therefore higher since the weights are no longer just associated with a variable, but with a variable conditionally to a class: w_{ik} is the weight associated to the variable i and the class k, b_k is the bias associated to the class k.

The first layer of our graphical model is a linear layer which realizes a weighted sum H_k for every class k, such as: $H_k = \sum_{i=1}^{d} w_{ik} log(P(X_i|C_k)) + b_k$. The second layer is a $Softmax$ such as: $P_k = \frac{e^{H_k}}{\sum_{j=1}^{K} e^{H_j}}$. Finally the graphical model, in the case where the inputs are based on the log conditional density estimation ($\log(p(X_i|C_k), \forall i, k)$, gives the values ($\forall k$) of the $P(C_k|X)$ such as:

$$P_k = \frac{e^{b_k + \sum_{i=1}^{d} w_{i,k} log(p(X_i|C_k))}}{\sum_{j=1}^{K} e^{b_j + \sum_{i=1}^{d} w_{i,j} log(p(X_i|C_j))}} \quad (2)$$

The input variables used as the inputs of this graphical model come from the on-line summaries described in section 3.

The optimization of the weights is done using a stochastic gradient descent for a given cost function. For a given example X the weights updates follow the formulae:

$$w_{ij}^{t+1} = w_{ij}^{t} - \eta \frac{\partial Cost^X}{\partial w_{ij}} \quad (3)$$

where $Cost^X$ is the cost function applied to the example X and $\frac{\partial Cost^X}{\partial w_{ij}}$ the derivative of the cost function w.r.t. the parameters of the model, here the weights w_{ij}.

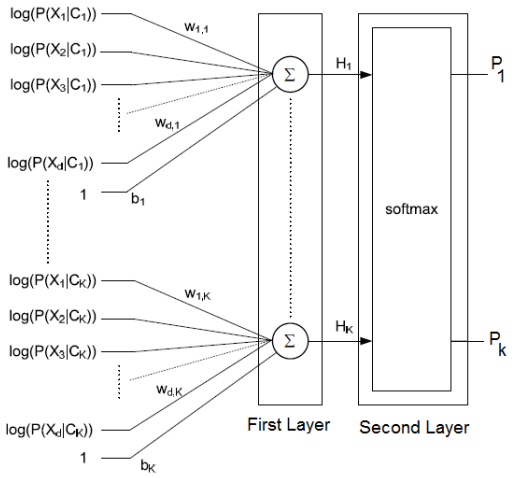

Fig. 1. Graphical model dedicated to the optimization of the weights.

The detail of this calculation is presented in the appendix of this paper, the result is simple:

$$\frac{\partial Cost}{\partial H_k} = P_k - T_k, \forall k \tag{4}$$

where T_k is the desired probability and P_k the estimated probability. So the update of the weights has a very low complexity.

The method used to update the weights is the one used usually for a standard back-propagation and three main parameters have to be considered (Lecun et al. (1998)) in the case of stochastic gradient descent: (i) the cost function; (ii) the number of iterations; (iii) the learning rate. For the cost function in the supervised classification context the best choice, since the output to learn takes only two values $\{0, 1\}$, is the log-likelihood (Bishop (1995)) which optimizes $log(P(C_k|X))$. The number of iterations in our case is set to 1 since each example of the stream is used only once for training. Finally the only parameter to set is the learning rate. A too small value will give a slow convergence but a high value may not allow reaching a global minimum. For off-line learning this value can be adjusted using a cross validation procedure but for on-line learning on data streams this procedure cannot be applied. In the experiments presented below the learning rate is fixed to $\eta = 10^{-4}$. However if we expect concept drift it could be interesting to have an adaptive learning rate as in (Kuncheva and Plumpton (2008)).

3 Conditional density estimation

This section presents how the conditional density probabilities used as inputs of our graphical model are estimated. This is not the main contribution of

this paper therefore the three methods used are briefly described. We use in the experimental part of this paper three methods (See Figure 2) to compute the conditional density probability for each numerical explanatory variable and for each target class: (i) our two layers incremental discretization method based on order statistics as described in (Salperwyck and Lemaire (2013)) (ii) a two layer discretization method "cPiD" which is a modified version of the PiD method of Gama (Gama and Pinto (2006)) (iii) and a Gaussian approximation. The approach could be the same for categorical variable (not detailed in this paper) by putting in the first level the count-min sketch approach and in the second level the grouping MODL method.

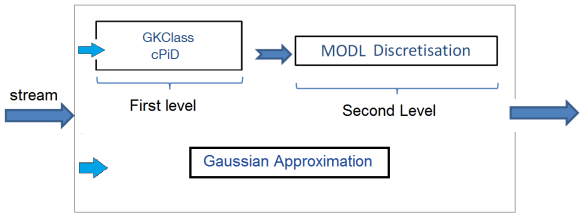

Fig. 2. On-line summaries for numerical variables: two level approach for GKClass+MODL and cPiD+MODL and single level approach for the Gaussian approximation.

The two level approach is based on a first level which provides quantiles which are order statistics on the data stream for every explanatory variable i. Each quantile, q, is a tuple which contains: $< v_q^i, g_q^i, (g_{qj}^i)_{j=1,...K} >$ where for each explanatory variable i of the data stream (i) v_q^i is a value (ii) g_q^i corresponds to the number of values between v_{q-1}^i and v_q^i (iii) (g_{qj}^i) is, for the K classes of the supervised classification problem, the number of elements in q belonging to the class j. The second level is a batch algorithm applied on the quantiles. In this paper the number of tuples is equal to 100 corresponding to the estimation of centiles. The tuning of the number of quantiles is discussed in (Salperwyck (2012)). cPid and GkClass, described below, are two different methods to obtained quantiles.

3.1 cPid

Gama and Pinto in (Gama and Pinto (2006)), proposed a two layer incremental discretization method. The first layer is a mix of a discretization based on the methods "Equal Width" and "Equal Frequency" (algorithm details: (Gama and Pinto (2006)) p. 663). This first layer is updated incrementally and needs to have much more bins than the second one. The second layer uses

information of the first one to build a second discretization. Many methods can be used on the second layer such as: Equal Width, Equal Frequency, Entropy, Kmeans... The advantage of this method is to have a fast first layer which can be used to build different discretizations on it (second layer).

In cPid we propose to change PiD to use a constant memory. In order to keep memory usage constant, after each split we merge the two consecutive intervals with the lowest counts sum. Thus the number of stored intervals remains constant. The second layer used is the MODL discretization (Boullé (2006a)). For brevity PiD and cPiD are not compared in this paper but interested readers can find this comparison in (Salperwyck (2012)).

3.2 GKClass

This algorithm proposed by Greenwald and Khanna (Greenwald and Khanna (2001) is an algorithm to compute quantiles using a memory of $O(\frac{1}{\epsilon}log(\epsilon N))$ in the worst case. This method does not need to know the size of the data in advance and is insensitive to the arrival order of the examples. The algorithm can be configured either with the number of quantiles or with a bound on the error. We adapted the GK summary to store directly the class counts in tuples. The second layer used is the MODL discretization (Boullé (2006a)).

The MODL discretization [5] and grouping are supervised and do not need any parameters. They are based on class counts and evaluate all possible intervals for the numerical variables and groups for the categorical variables. The quality evaluation of the model is based on a Bayesian approach. For numerical variables, its purpose is to find the best discretization parameters: number of intervals, frontiers of the intervals and class distribution in the intervals in a Bayesian sense. For categorical variables the method performs grouping in a similar way.

3.3 Gaussian approximation

The main idea of this method relies on the hypothesis that the observed data distribution follows a Gaussian law. Only two parameters are needed to store a Gaussian law: its mean and its standard deviation. The incremental version requires one more parameter: the number of elements. This method has one of the lowest memory footprints and is constituted of a single layer but is dependent on the Gaussian assumption. It will be used as a reference method as the Large Scale Learning challenge datasets are generated using a Gaussian generator.

4 Experiments

4.1 Protocol

The Large Scale Learning challenge datasets (organized by the network of excellence PASCAL, `http://jmlr.csail.mit.edu/papers/topic/large_scale_`

`learning.html`) have been used for the experiments. These datasets are constituted of 500,000 labeled examples which is large enough to evaluate an on-line classifier. The datasets alpha, beta, delta and gamma contain 500 numerical explanatory variables and the dataset epsilon and zeta 2000 numerical variables. The 100,000 first examples have been used as the test dataset and the remaining examples as the train dataset.

4.2 Results

For the first part of the experiments, a standard naive Bayes (without the weighting) is used on the top of the summaries described in section 3. The results are presented in table 1 and shows that the estimation of the conditional probabilities is accurate for all methods used since all the naive Bayes classifiers obtain good results. Despite the fact that the large scale learning datasets come from Gaussian generators the two others summaries obtain similar results without the Gaussian assumption. One additional advantage of the method based on GKClass summaries (level 1) with the MODL discretization (level 2) is that it guarantees a maximal error for a given memory on the quantile estimation. Therefore this two level method is used for the second part of the experiments.

	Alpha			Beta			Delta		
#train examples →	40 000	100 000	380 000	40 000	100 000	380 000	40 000	100 000	380 000
GKClass	54,40	54,49	54,56	49,79	51,05	51,23	80,56	82,22	83,47
CPiD	54,36	54,52	54,57	49,79	51,09	51,12	80,66	82,39	83,77
Gaussien (niveau 1)	54,62	54,67	**54,67**	51,21	**51,50**	51,31	84,58	**85,10**	85,08
	Gamma			Epsilon			Zeta		
#train examples →	40 000	100 000	380 000	40 000	100 000	380 000	40 000	100 000	380 000
GKClass	92,63	93,51	94,23	70,48	70,37	70,43	78,35	78,63	78,48
CPiD	92,93	93,83	94,41	**70,52**	70,38	70,36	78,50	78,48	78,42
Gaussian	95,09	95,10	**95,16**	70,66	70,57	70,43	**78,96**	78,77	78,52

Table 1. Accuracy of a Naive Bayes (without Averaging) using the three methods to compute the conditional probabilities.

Knowing the results of the table 1, we know that the estimation of the conditional probabilities is accurate. Therefore the behavior of our Weighted Naive Bayes can be studied. The results are obtained using 4 classifiers: (1) a naive Bayes (NB) trained offline with the MODL discretization (Boullé (2006a)) and all the data in memory; (2) an Averaged Naive Bayes (ANB) trained offline with the MODL discretization (Boullé (2006a)), this algorithm is described in (Boullé (2006b)) to compute the weights and all the data in memory - this method is one of the best of the literature see (Guyon et al. (2009)) (3) a Naive Bayes trained online with the two level discretization method which uses GKClass (level 1) and the MODL discretization (level 2)

4) a Weighted Naive Bayes trained online with the two level discretization method which uses GKClass (level 1) and the MODL discretization (level 2) and our method based on the graphical method to estimate the weights.

	Alpha			Beta			Delta		
#train examples→	40 000	100 000	380 000	40 000	100 000	380 000	40 000	100 000	380 000
Offline NB (1)	54,60	54,61	54,61	49,79	51,36	51,19	80,78	82,44	83,91
Offline ANB (2)	66,13	67,30	*68,77*	49,79	51,39	*53,24*	80,80	82,46	*83,91*
Online NB (3)	54,40	54,49	54,56	49,79	51,05	51,23	80,56	82,22	**83,47**
Online ANB (4)	64,03	66,40	**67,61**	49,79	49,79	**52,20**	75,35	77,74	79,53

	Gamma			Epsilon			Zeta		
#train examples →	40 000	100 000	380 000	40 000	100 000	380 000	40 000	100 000	380 000
Offline NB (1)	92,95	93,99	94,63	70,35	71,04	70,58	78,39	78,34	78,26
Offline ANB (2)	92,95	94,00	*94,64*	84,36	85,34	*86,01*	88,67	89,51	*90,43*
Online NB (3)	92,63	93,51	**94,23**	70,48	70,37	70,43	78,35	78,63	78,48
Online ANB (4)	90,25	91,05	91,76	69,25	74,76	**79,95**	73,82	79,91	**84,52**

Table 2. Comparison of the Accuracy of Naive Bayes (NB), the averagated Naive Bayes (ANB) and the weighted Naive Bayes (WNB) . The best 'offline-result' is in italics and the best 'online result' is bold.

The table 2 shows that the results obtained by our On-line WNB are very interesting: it is better than the on-line NB except for the Gamma and Delta datasets. Its performance is close to the Off-line ANB which is the "best results" which can be obtained on these datasets with Bayes type classifiers. But the classifier makes the assumption that all the data are stored in memory and readable several times whereas this is not possible on data streams. Our WNB approach uses a low amount of memory thanks to the two level approach to estimate the conditional densities, and is purely incremental thanks to the graphical model and the stochastic gradient descent to estimate the weights. Therefore the results of our approach are very promising. The accuracy obtained versus the number of examples used to train the model seems to indicate that our on-line WNB (4) and the off-line ANB (2) would have the same accuracy if the number of examples was higher. The dimension could also be an explanation (Alpha, Beta, Delta and Gama have 500 numerical variables, Epsilon and Delta have 2000 variables) since the accuracy obtained by Online WNB increases significantly with the number of examples for Epsilon and Zeta. These points will be investigated in future works.

5 Conclusion

The results for our on-line weighted version of the Naive Bayes classifier are promising. It improves the performance compared to the non-weighted version and is close to the off-line averaged version of the naive Bayes classifier. However we think that its results could be improved in future works. The first

idea would be to use the GK Class summaries as "mini-batch" (Cotter et al. (2011)) and do several iterations to speed up the gradient descent. The second idea would be to use an adaptive learning rate: high at the beginning and low after, or to take into account the error rate as in (Kuncheva and Plumpton (2008)). Future works will be done in these directions.

References

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, USA.

Boullé, M. (2006a). MODL: A Bayes optimal discretization method for continuous attributes. *Machine Learning*, 65(1):131–165.

Boullé, M. (2006b). Regularization and Averaging of the Selective Naive Bayes classifier. *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1680–1688.

Cotter, A and Shamir, O. And Srebro N and Sridharan, K. (2011) Better Mini-Batch Algorithms via Accelerated Gradient Methods. *CoRR*.

Gama, J. and Pinto, C. (2006). Discretization from data streams: applications to histograms and data mining. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 662–667.

Greenwald, M. and Khanna, S. (2001). Space-efficient online computation of quantile summaries. *ACM SIGMOD Record*, 30(2):58–66.

Guigourès, R. and Boullé, M. (2011). Optimisation directe des poids de modèles dans un prédicteur Bayésien naif moyenné. In *Extraction et gestion des connaissances EGC'2011*, pages 77–82.

Guyon, I., Lemaire, V., Boullé, M., Dror, G., and Vogel, D. (2009). Analysis of the KDD Cup 2009: Fast Scoring on a Large Orange Customer Database. *JMLR: Workshop and Conference Proceedings*, 7:1–22.

Hoeting, J., Madigan, D., and Raftery, A. (1999). Bayesian model averaging: a tutorial. *Statistical science*, 14(4):382–417.

Koller, D. and Sahami, M. (1996). Toward Optimal Feature Selection. *International Conference on Machine Learning*, 1996(May):284–292.

Kuncheva, L. I. and Plumpton, C. O. (2008). Adaptive Learning Rate for Online Linear Discriminant Classifiers. In *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 510–519. Springer-Verlag.

Langley, Pat and Iba, W. and Thompson, K. (1992). An analysis of Bayesian classifiers. In *Proceedings of the National Conference on Artificial Intelligence*, pages 223-228.

Langley, P. and Sage, S. (1994). Induction of Selective Bayesian Classifiers. In Poole, R. L. D. M. and D, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406. Morgan Kaufmann.

Lecun, Y., Bottou, L., Orr, G. B., and Müller, K. R. (1998). Efficient BackProp. In Orr, G. and Müller, K., editors, *Lecture Notes in Computer Science*, volume 1524 of *Lecture Notes in Computer Science*, pages 5–50. Springer Verlag.

Salperwyck, C. (2012). *Apprentissage incrémental en ligne sur flux de données*. PhD thesis, University of Lille.

Salperwyck, C. and Lemaire, V. (2013). A two layers incremental discretization based on order statistics. *Statistical Models for Data Analysis, 2013, pp 315-323.*

Appendix - Derivative of the Cost Function

The graphical model is built to have directly the values of the $P(C_k|X)$ at the output. The goal is to maximize the likelihood and therefore to minimize the negative log likelihood. The first step in the calculation is to decompose the softmax considering that each output could be seen as the succession of two steps: an activation followed by a function of this activation.

Here the activation function could be seen as: $O_k = f(H_k) = exp(H_k)$ and the output of the softmax part of our graphical model is: $P_k = \frac{O_k}{\sum_{j=1}^{K} O_j}$. The derivative of the activation function is:

$$\frac{\partial O_k}{\partial H_k} = f'(H_k) = exp(H_k) = O_k \tag{5}$$

The cost function being the $-log\,likelihood$, we have to consider two cases: (i) the desired output is equal to 1 or (ii) the desired output is equal to 0. For the following we note:

$$\frac{\partial Cost}{\partial H_k} = \frac{\partial C}{\partial P_k}\frac{\partial P_k}{\partial O_k}\frac{\partial O_k}{\partial H_k} \tag{6}$$

In the case where the desired output of the output k is equal to 1 by replacing (5) in (6):

$$\frac{\partial Cost}{\partial H_k} = \frac{\partial C}{\partial P_k}\frac{\partial P_k}{\partial O_k}\frac{\partial O_k}{\partial H_k} = \frac{-1}{P_k}\frac{\partial P_k}{\partial O_k}O_k \tag{7}$$

$$\frac{\partial Cost}{\partial H_k} = \frac{-1}{P_k}\left[\sum_{l=1,l\neq k}^{K}\left(\frac{O_l}{(\sum_{j=1}^{K} O_j)^2}\right)\right]O_k = \frac{-1}{P_k}\left[\frac{(\sum_{j=1}^{K} O_j) - O_k}{(\sum_{j=1}^{K} O_j)^2}\right]O_k \tag{8}$$

$$\frac{\partial Cost}{\partial H_k} = \frac{-1}{P_k}\left[\frac{(\sum_{j=1}^{K} O_j) - O_k}{(\sum_{j=1}^{K} O_j)}\right]\frac{O_k}{(\sum_{j=1}^{K} O_j)} = \frac{-1}{P_k}\left[1 - \frac{O_k}{(\sum_{j=1}^{K} O_j)}\right]\frac{O_k}{(\sum_{j=1}^{K} O_j)} \tag{9}$$

Therefore

$$\frac{\partial Cost}{\partial H_k} = \frac{-1}{P_k}[1 - P_k]P_k = P_k - 1 \tag{10}$$

In the case where the desired output of the output k is equal to 0 the error is only transmitted by the normalization part of the $softmax$ function since the derivative for an output where the desired value is 0 is equal to 0. Therefore with similar steps we have: $\frac{\partial Cost}{\partial H_k} = P_k$

Finally we conclude: $\frac{\partial Cost}{\partial H_k} = P_k - T_k, \forall k$ where T_k is the desired probability and P_k the estimated probability. Then the rest of the calculation of $\frac{\partial Cost}{\partial w_{ik}}$. is straightforward.