# A model for the labeler in active learning: Accounting for "I don't know" and "not this"

A1, A2

October 25, 2013

**Abstract**

We discuss the active learning of a classifier when 1) a labeler returns a probability vector over the labels 2) the labeler's certainty of the labels is not constant over the item space. We show how the labeler uncertainty can be modelled using Gaussian process latent variable model, and how incorporating the uncertain labeler model to the optimal sampling scheme improves classification under uncertain labeling.

## 1 Introduction

The optimal design of sampling for improving model estimates, also known as active learning, is a popular method for training classifiers on large collection of items. In short, an unlabelled item database is ranked based on the usefullness of the expected label for the current state of the classifier. Then an external human labeller is asked for the correct label and the model is re-estimated. In theory, the learning rate of the classifier is optimal as the data so collected is the most useful.

The process relies on the labeller ability to produce the correct label, hence the often used name 'oracle'. The oracle is assumed all-knowing, and several approaches have been introduced to alleviate this unrealistic assumption. In general, the label is allowed to be wrong with some probability, reflecting e.g. the limited knowledge of the labeler.

The five papers studying labellers with imperfect answers [][][][][] assume the following: ...

In this paper we discuss two novel ideas. First, we want to accommodate the answers "definitely not this one" and "I don't know". Second, we want to account for the user's limited knowledge in the active learning process.

## 2 A model for flexible answers

The core problem is the training of a classifier in a semi-supervised manner. We have a set of labeled training items, and we want to improve the classifier by querying an external labeler for some more labels for unlabeled items. The idea is that asking is expensive, so we should only query the most useful items.

(Recap: oracle, noisy oracle, Yan++.)

Formally, let $\mathcal{D} = \{i = (y_i, x_i)\}$ denote a set of items, where $y_i \in \mathcal{Y}$ with $|\mathcal{Y}| = K$ denote labels for the items and $x_i \in \mathcal{X} \subseteq \mathbb{R}^p$ denote other features. Write for any $x \in \mathcal{D} \cap \mathcal{X}$ the correct label $y_x$.

We assume the pool-based sampling scheme, i.e. we have a pool $\mathcal{U} \subset D \cap \mathcal{X}$ of unlabeled items which we can use for querying the labeler for their labels. We assume that the cost of querying are constant.

Assume that we start training the classifier with a set of correcly labeled items $\mathcal{L}_0 \subset D$, where the classifier can be written as

$$y|x,\theta \sim p(y|x,\theta) = \prod_k p(k|x,\theta)^{[y=k]} \quad (1)$$

$$\theta \sim p(\theta) \quad (2)$$

with $\theta$ being the parameters of the model, and $p(\theta)$ denoting a prior distribution for the parameters. The simple active learning scenario assumes that we have access to an oracle labeller which provides the $y_x$ for any $x \in U$ we query, providing us more correctly labelled data.

The oracle assumption is not realistic when the labeller is human, so uncertainty model should be incorporated. Our approach is to assume that for each query "what is the label of $x$?", the labeller returns a probability vector $h_x \in \mathcal{H} = \{h \in \mathbb{R}_+ : \sum_{k \in \mathcal{Y}} h_k = 1\}$. We furthermore assume that the labeller is biased towards the correct label, i.e. $\text{argmax}_k h_{xk} = y_x$.

The oracle labeller is the special case "definitely $k$" with $h = \mathbf{1}\{h_k = 1\}$. The "I don't know" case is $\mathbf{1}\{h_k = K^{-1}\ \forall k\}$, and in between are a continuum of alternatives, including the "definitely not $k$" with $\mathbf{1}\{h_k = 0\}$ and "one of these $M$" with $\mathbf{1}\{h_k > 0\ \forall k \in M\}$.

The connection between $h$ and $y$ needs to be made in order for the extension to be useful in learning the classifier parameters $\theta$. We assume that the following model is valid:

$$p(h|x,\theta) \propto p(h|x) \prod_k p(k|x,\theta)^{h_k} \quad (3)$$

in which the $y$ variable is no longer necessary to be observed. The model is a continuous version of categorical model, and as $y$ is a categorical variable, we can replace the observations $y$ with indicator vector $1(h_y = 1)$. An interpretation for the approach is given by the multinomial distribution: the product is approximately the likelihood of a multinomial variable if $h$ would be integers. The scaling of $h$ is not important, the pairwise proportions of $h_k$ are the main information. As a practical example, the user could be given a set of ten tokens to distribute among the label candidates.

To see whether the approach is useful we conducted a simulation study. We chose the base classifier to be the Naive Bayes classifier with Gaussian distributions. For a set of observations $L = \{(h_i; x_i)\}$ the ML estimate for $\theta = \{\mu_k, \Sigma_k : k = 1, ..., K\}$ are given as weighted averages and covariance matrices, i.e. each queried item $x_i$ is 'observed' in each of the $K$ classes with a weight $h_i$.

For the item set $D$ we simulated three 2D Gaussian clusters, 50 points each. Two points from each cluster were given as the initial data for the classifier. The imitation human labeller knew one of the classes, say $k'$, returning an indicator vector for items of class $k'$, and when queried an item of other classes he returns 0.5 for $k \neq k'$ (0 for $k'$ as he knows the label is not $k'$). For comparison we forced the user to make a decision, simulated here by a random pick when he did not know the label.

We ran 20 different samplings sequences from the item set, each step drawing uniformly from the pool of unlabeled items. The classification rate over a test set for the forced-to-choose model were averaged over 10 random picking sequences per sampling, estimating the expectation over the forcing event. Figure 1 shows the $(0.05, 0.5, 0.95)$ quantile envelope plots of the classification rate a function of sample size.

It is quite clear that the learning is improved if the user is allowed to express his uncertainty and not forced to choose a single value. But how does this aid in the active learning scenario? By letting us estimate the labeller's uncertainty.

# 3 The answer dependends on the question

Assume that $\alpha(x) \in \mathbb{R}$ is a value that describes the uncertainty the labeller has on the label of $x$, and assume that $\alpha$ is continuous and varies smoothly in $\mathcal{X}$.

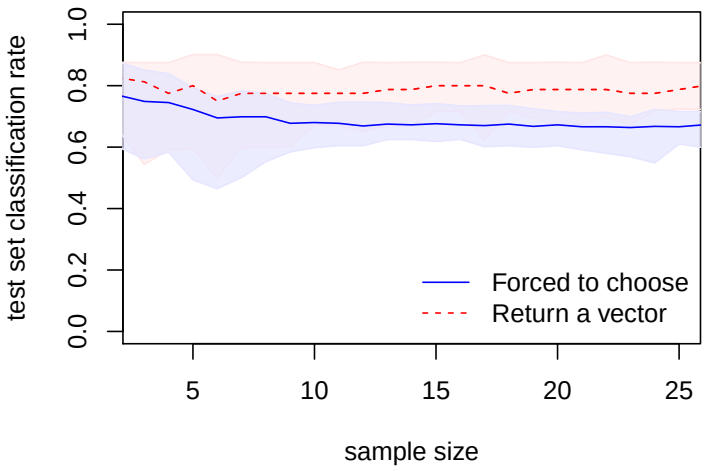


Figure 1: Classification rate example, two modes the user can provide labeling information: Choose one vs. a weight vector. The shaded area is the 90% range over data sampling sequences, 20 different samplings. The forced-to-choose values are further averaged over 10 random selection sequences per sampling sequence.

Continuing from the Equation 3, we now introduce a prior for the $h$ values. We model them as dependent on the smooth $\alpha$, which we assume in this paper to be adequately described by a log-Gaussian process in $\mathcal{X}$. The natural model for $h$ is the Dirichlet distribution, we simplify the problem by assuming symmetry, i.e. the Dirichlet parameter $\alpha$ is the same for all classes. The interpretation is that high value of $\alpha$ means high uncertainty, and vice versa.

The prior $p(h|x, \alpha)$ for the labeller's variable output then becomes

$$h|x, \alpha \sim Dir(\alpha(x)) \quad (4)$$

$$\alpha \sim \log -GP(\mu, C) \quad (5)$$

with some hyper-prior mean $\mu$ and covariance function $C$. In what follows the parameter $\alpha$ shall be known as the *labeller's uncertainty*.

As an example we simulated a labeller with a circular "area of expertice", a disc in the feature space wihtin whic he knows the labels. Outside he returns "I don't know". Figure 2 depicts the Kriging surface estimate of $\alpha$ after 10 and 20 uniformly chosen queries. Note how the area of expertice is being revealed.

For optimal result the classifier itself would account for the labeller's uncertainty, but unfortunately the hierarchical nature of the labeller model is not trivial to implement. For simplicity of implementation we will focus our discussion on the case where the labeller model is independent from the classifier model. Immediate question arises: How can we combine the two models in order to do active learning?

# 4 Active learning with uncertain labellers

For illustration we shall discuss the popular *uncertainty sampling* strategy for active learning: At any time point the model parameter estimates are used for computing an informativeness function $I(x) = I(x; \hat{\theta})$,

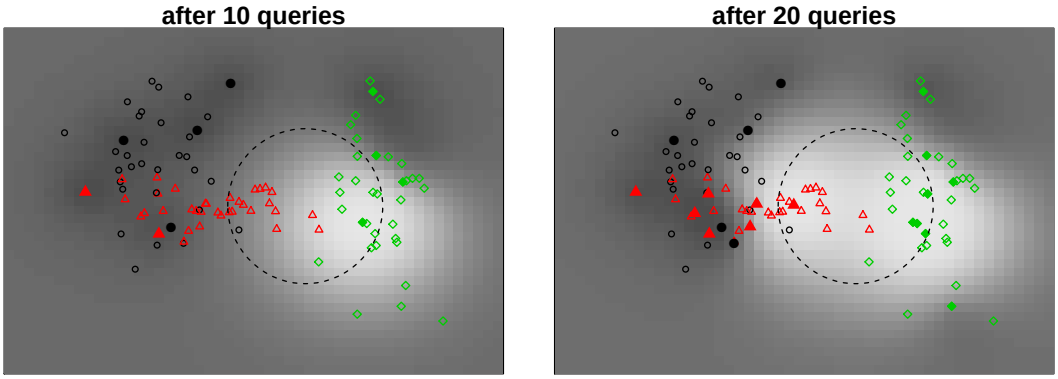


Figure 2: Labeller's uncertainty as estimated after 10 (left) and 20 (right) queries. Symbols denote the item set, filled symbols indicate items that were queried. Dashed disc is the labeller's area of expertice where he knows the label. Dark color of the background is for low certainty areas and light color is for high certainty areas of the feature space.

providing a measure of usefullness of gaining the label of an item $x \in \mathcal{U}$. There exists several formulations of $I(x)$, but we will focus on the entropy,

$$I_c(x) := -E_y[\log p(y|x, \hat{\theta})] \quad (6)$$

$$= -\sum_k p(k|x, \hat{\theta}) \log p(k|x, \hat{\theta}) \quad (7)$$

where the probabilities are the posterior predictive probabilities at the current iteration. The uncertainty sampling strategy then queries the item which maximizes the informativeness $I_c$.

Now, the uncertainty sampling assumes that the labeller is an oracle, providing us the label that reduces the entropy the most. But as the uncertain labeller does not always know the label, the expected gains are reduced. The informativeness given by the labeller can also be expressed using the entropy,

$$I_l(x) := -E_h \log p(h|x, L) \quad (8)$$

where $L$ is all so far collected data. The entropy of a Dirichlet distributed $h$ depends on the $\alpha$, leading to the formula

$$I_e(x) = E\{\log B(\alpha(x)) + K(\alpha(x) - 1)[\psi(K\alpha(x)) - \psi(\alpha(x))]\} \quad (9)$$

where $B$ is the beta-function, $\psi$ is the digamma function, and the expectation is over $\alpha(x)$. Unfortunately this function is not monotonous in $\alpha$, as the $h$ will converge to uniform as $\alpha$ increases, and the entropy decreases.

Uniform $h$ should be regarded as the least informative answer from the labeller. We therefore suggest the use of variance which is a monotonically decreasing function of $\alpha$,

$$I_l(x) := Var[h|x, \alpha] = \frac{K - 1}{K^2(K\alpha(x) + 1)}.$$

The two $I$ functions do not share units, and to overcome this we use the geometric mean to arrive at the *labeller adjusted informativeness*

$$I(x) := (I_l(x) I_c(x))^{1/2}$$

which is the basis of *labeller adjusted uncertainty sampling* strategy for active learning.

We applied the adjusted ucertainty sampling strategy to the example of circular area of expertice. The lower left plot of Figure 3 depicts the learned labeller's uncertainty after 25 queries, along with the posterior 95% probability ellipses of the clusters. The top left and top right plots show what happends when only $I_l$ and $I_c$, respectively, are used. If only those items are queried of which the user is certain, it will result in greedy sampling of items close to known good quality points, regardless of their utility to the classifier. If only the classifier is concerned, the lack of information of items outside labeller's area of expertise is not realized. The use of both provides a compromise, and can lead to substantial improvement as depicted in the test set classification rates. For the joint strategy the rate reaches almost the rate achieved when total knowledge of the labels is available, marked with the horizontal line in the bottom right plot.

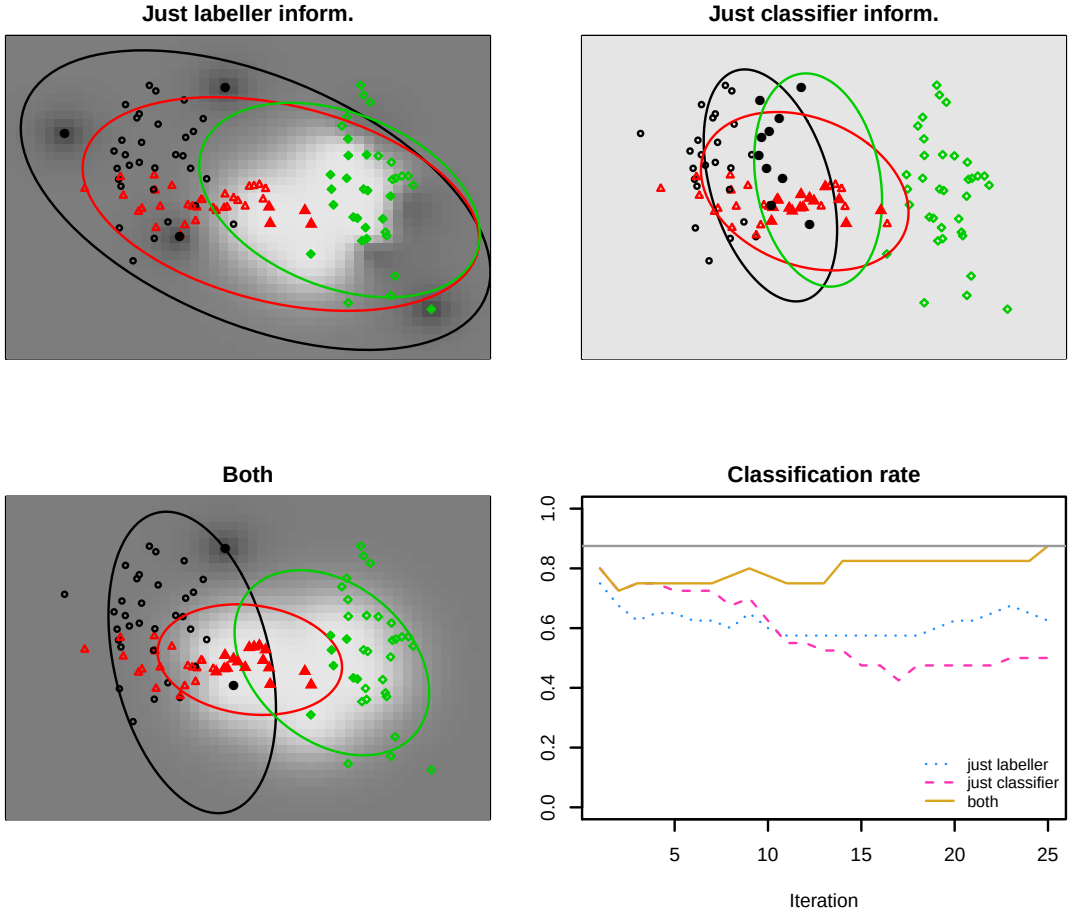


Figure 3: Three results after 25 steps depending on which query strategy was used, and their classification rates on a test set as a function of steps. The horizontal line is for full training data with an oracle.

# 5 Application example