Approximation algorithms for reducing classification cost in ensembles of classifiers

Sarah R. Allen*
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
srallen@cs.cmu.edu

Lisa Hellerstein
Department of Computer Science and Engineering
Polytechnic Institute of NYU
Brooklyn, NY 11201
hstein@poly.edu

Abstract

We develop approximation algorithms for reducing expected classification cost, when there is a cost associated with obtaining the value of each attribute, and we are doing classification based on an ensemble of linear threshold classifiers. We focus on the stochastic setting where attribute values are independent, and their distributions are given. We review related work based on reductions to Stochastic Submodular Set Cover. We prove approximation bounds for determining the majority prediction of the classifiers in the ensemble, and for detemrining the prediction of at least one classifier.

1 Introduction

Consider the problem of reducing classification cost, when there is a cost associated with obtaining the value of each attribute. We adaptively choose which attributes to pay for, until we determine the output of the classifier. This problem has been studied from both the theoretical and experimental standpoint in a number of recent papers(cf. [2, 3]). We assume here that the classifier has already been learned, and that we are trying to reduce classification cost in a post-processing stage. We assume a stochastic setting where the attributes are independent, and the distribution for each attribute is given as input. The goal is to minimize the expected classification cost. This version of the cost-minimization problem has been studied in machine learning in the context of learning with attribute costs [4], in operations research under the name "sequential testing", and in other fields (cf. [5]).

We focus on the problem of minimizing the expected cost of classification when using ensembles of linear threshold classifiers. We assume that each attribute only has a constant number of possible values.[1]

Even if there is only a single linear threshold classifier that we need to evaluate, the cost minimization problem is NP-hard [6]. We are interested in developing good approximation algorithms. The naïve algorithm of testing each variable in order of increasing cost obtains an approximation ratio of n, so we consider such a factor to be trivial. In recent work, Deshpande et al. gave a poly-time approximation algorithm for mimimizing the cost of evaluating a single linear threshold classifier [7]. It is a *3-approximation* algorithm, meaning its expected cost that is at most a factor of 3 greater than the minimum possible expected cost. They also considered the problem of minimizing the cost of

*Preliminary versions of some of these results appeared in the thesis submitted to the Faculty of the Polytechnic Institute of New York University in partial fulfillment of the requirements for the degree of Master of Science (Computer Science), May 2013 [1].

[1] Many of the results discussed also hold if each attribute is distributed according to a discretized version of a continuous distribution, with support in a bounded interval, and the cdf of the distribution can be computed in polynomial time at any point.

evaluating m linear threshold classifiers in an ensemble when the classifiers can share attributes and the goal is to determine the output of each of the classifiers.

In what follows, we will review the techniques used in the above results, and apply them to further problems involving ensembles of m linear threshold classifiers. In particular, we consider the problem of minimizing the cost of determining the following:

- The majority vote, or a weighted majority vote, of the m classifiers
- The value of at least one of the m classifiers.
- A common value that is predicted by at least $z \leq \lceil \frac{m}{2} \rceil$ of the classifiers.

The first problem is standard. The second problem is not, but it is potentially important in the context of reducing classification costs. The m classifiers may, for example, be produced by m different learning algorithms, all reasonably accurate. If the classifiers in the ensemble are defined over different subsets of variables, there is potentially a big cost savings in stopping testing as soon as the value of one of these classifiers is known. The first problem is a case of the third problem where $z = \lceil \frac{m}{2} \rceil$. We present the first approximation algorithms for these problems, with approximation factors that are independent of the number of variables n, and the coefficients of the classifiers. However, the factors depend linearly on m, the number of classifiers. An open question is to develop approximation algorithms for these problems having better approximation factors.

There are also many other open questions. One is whether better approximation factors can be obtained by relaxing the requirement that predictions must always be computed exactly. A related quetion is whether approximation results can be obtained for procedures that integrate learning with classification-cost reduction, rather than doing the cost reduction as a post-processing step.

2 Background

We give an overview of the techniques used by Deshpande et al. in [7] and then present some technical background. The 3-approximation algorithm for linear threshold evaluation is based on a reduction from linear threshold evaluation to a Stochastic Submodular Set Cover (SSSC) problem, defined below, through the construction of an appropriate utility function. The SSSC problem is then solved using Adaptive Dual Greedy (ADG), an algorithm for SSSC developed by Deshpande et al.

The SSSC problem can also be solved using another algorithm, the Adaptive Greedy (AG) algorithm of Golovin and Krause. If we replace ADG by AG in the above linear threshold evaluation algorithm, the resulting approximation factor is not 3, but $O(\log D)$, where D is the sum of the magnitudes of the coefficients in the linear threshold formula (for simplicity, we assume all coefficients are integers). In this context, ADG is a better choice than AG.

When dealing with problems involving *ensembles* of linear threshold classifiers, though, AG is sometimes the better choice. For the problem of evaluating all m linear threshold classifiers in an ensemble, using AG instead of ADG yields an $O(\log(mD_{avg}))$-approximation algorithm, where D_{avg} is the average over all m functions of the sum of the magnitudes of the function's coefficients [7]. ADG yields a worse bound in this case.

We now give formal definitions. For simplicity of exposition, we will assume that attributes are binary-valued.

In the *linear threshold evaluation* problem, we need to compute the value of a linear threshold formula h over the binary attributes $x_1, \ldots, x_n$, on an unknown input $x = (x_1, \ldots, x_n)$. The formula h has the form $a_1x_1 + \ldots + a_nx_n \geq \theta$. We assume without loss of generality that all a_i, and threshold θ are integers. For each x_i, we are given as input the cost c_i of determining the value of that attribute, and the probability p_i that $x_i = 1$, where $0 < p_i < 1$ and $c_i > 0$. When we determine the value of attribute x_i, we say that we are *testing* that x_i. Testing is performed in an on-line fashion, and the choice of the next test can depend on the outcomes of previous tests. Testing continues until the value of formula h on x is computed and output. The problem is to adaptively determine the order of the tests and compute the output, so as to minimize the total expected testing cost. Given an algorithm for this problem, the running time is the worst-case time to compute the next test to perform.

To solve linear threshold evaluation problems, we construct a utility function g which measures the value of the information obtained from the tests performed so far. We use a partial assignment $b \in \{0,1,*\}^n$ to denote the information, where $b_i = *$ when the value of x_i is unknown, and b_i is the value of x_i otherwise. Utility function $g : \{0,1,*\}^n \mapsto \mathcal{N}$ assigns a value to each such g.

For $c, b \in \{0,1,*\}^n$, we say that c *extends* b, written $c \sim b$, if $c_i = b_i$ for all i where $b_i \neq *$. Additionally, let $b_{i \leftarrow \ell}$ denote the partial assignment in which b_i is assigned to value $\ell \in \{0,1\}$ and all other b_j remain unchanged.

Given utility function $g : \{0,1,*\}^n \mapsto \mathcal{N}$, and a value $Q \in \mathcal{N}$, we call g a *submodular cover function* with goal value Q if the following properties hold:

1. $g(*, \ldots, *) = 0$
2. $g(a) = Q$ for all $a \in \{0,1\}^n$
3. (monotonic) $g(b) \leq g(c)$ for all $b, c \in \{0,1,*\}^n$ with $c \sim b$
4. (submod.) Given c, b, i s.t. $c \sim b$ and $c_i, b_i = *, \forall_{\ell \in \{0,1\}}, g(b_{i \leftarrow \ell}) - g(b) \geq g(c_{i \leftarrow \ell}) - g(c)$.

We use the terms monotonicity and submodularity here in an extended sense, applying them to functions defined on partial assignments (equivalently, on subsets of items and the states of those items).

The (binary) Submodular Set Cover (SSSC) problem is similar to the linear threshold evaluation problem. There is an initially unknown input $x \in \{0,1\}^n$, for each x_i, we are given as input the cost c_i of determining the value of x_i, and the probability p_i that $x_i = 1$ ($0 \leq p_i \leq 1$ and $c_i > 0$). Instead of being given a formula to evaluate, we are given a submodular cover function $g : \{0,1\}^n \mapsto \mathcal{N}$ with goal value Q, and we need to do enough tests so that the information acquired, represented by $b \in \{0,1,*\}^n$, satisfies $g(b) = Q$. Again, the problem is to determine how to adaptively choose the tests so as to minimize expected testing cost.

Golovin and Krause presented the Adaptive Greedy (AG) algorithm for the SSSC problem, and showed it was a $(\ln Q + 1)$-approximation algorithm [8]. Deshpande et al., using a different analysis, proved that it is a $2(\ln P + 1)$-approximation algorithm for binary x_i, where P is the maximum amount of utility achievable from a single test. (It is a $k(\ln P + 1)$-approximation algorithm for k-ary x_i.)

The Adaptive Dual Greedy (ADG) of Deshpande et al. is an α-approximation for the SSSC problem, where α is the maximum value of the ratio $\max \frac{\sum_{j \in \delta(C,b)} g(b_{j \leftarrow C_j}) - g(b)}{Q - g(b)}$ where the max is taken over all pairs (C, b) such that C is the partial assignment representing the tests performed by running ADG on some $a \in \{0,1\}^n$, b is a partial assignment where $C \sim b$, and $\delta(C, b) = \{j : C_j \neq *, b_j = *\}$.

Construction of Utility Functions

We review techniques for constructing the utility functions needed to reduce threshold evaluation problems to SSSC problems. We construct the utility functions by expressing them as the disjunction and/or conjunction of (submodular cover) utility functions.

In the disjunctive construction, we take two submodular cover functions, g_0 and g_1 defined on $\{0,1\}^n$, with goal values Q_0 and Q_1, and form a new new submodular cover function g with goal value $Q = Q_0 Q_1$, such that $g(b) = Q_0 Q_1 - (Q_0 - g_1(b))(Q_1 - g_1(b))$. Thus $g(b) = Q$ iff $g_1(b) = Q_1$ or $g_0(b) = Q_0$. In the conjunctive construction, $g = g_1 + g_0$ and the goal value is $Q = Q_1 + Q_0$. Thus $g(b) = Q$ iff $g_1(b) = Q_1$ and $g_0(b) = Q_0$. In both constructions, if g_0 and g_1 are submodular cover functions, so is g. These constructions easily extend to combining $k > 2$ cover functions $g_1, \ldots, g_k$. These constructions have appeared in a number of previous papers, either implicitly or explicitly [9–11].

It will be helpful to note the following about the above definition of g, in the disjunctive case. Formally, for $g_1, \ldots, g_k$, with goals $Q_1, \ldots, Q_k$, the utility function produced by the disjunctive construction is $g(b) = \prod_{j=1}^{k} Q_j - \prod_{j=1}^{k} (Q_j - g_j(b))$ with goal value $\prod_{j=1}^{k} Q_j$. The value of this $g(b)$ on a partial assignment b, can be associated as follows with a complete k-partite hypergraph. The hypergraph has $Q_1 + \ldots + Q_k$ vertices, and the vertices are partitioned in k sets, one for each g_j.

There are Q_j vertices in the set for g_j. Each of the $\prod_{j=1}^{k} Q_j$ hyperedges is a set consisting of one vertex from each of the k sets. The goal value for g is the total number of hyperedges. For partial assignment b, $g(b)$ is the number of hyperedges deleted from the graph if for each i, we remove $g_i(b)$ vertices from the set of vertices corresponding to g_i, and delete all hyperedges containing those vertices.

In what follows, we will make repeated use of a utility function g, constructed for a single linear threshold formula h, presented in [7]. For simplicity here, we will give the definition of g for the special case in which the coefficients of h are non-negative. The extension to arbitrary integer coefficients is not difficult, and is presented in [7].

Let $f : \{0,1\}^n \mapsto \{0,1\}$ be a linear threshold function represented by the formula $\sum_{i=1}^{n} a_i x_i \geq \theta$ such that $f(x) = 1$ if $\sum_{i=1}^{n} a_i x_i \geq \theta$ and $f(x) = 0$ otherwise. Assume $a_i \geq 0$ for all x_i. For $b \in \{0,1,*\}^n$, let $g_1(b) = \max\{\sum_{i:b_i=1} a_i, \theta\}$, and $g_0(b) = \max\{\sum_{i:b_i=0} a_i, D - \theta + 1\}$. Let $D = \sum_i |a_i|$. Let $Q_1 = \theta$ and let $Q_0 = D - \theta + 1$. It is easy to verify that if b represents the information from tests performed so far, then $g_1(b) = Q_1$ iff b has a 1-certificate for f (that is, $f(b') = 1$ for all $b' \sim b$), and $g_0(b) = Q_0$ iff b has a 0-certificate for f (that is, $f(b') = 0$ for all $b' \sim b$).

Applying the disjunctive construction described above to g_0 and g_1 produces a submodular cover function g with goal value $Q = Q_0 Q_1 \leq D(D+1)$ such that $g(b) = Q$ iff b has either a 0-certificate or a 1-certificate for f. Using the $(\ln Q + 1)$ bound on AG, we immediately get an $O(\log D)$ approximation algorithm for evaluating a single linear threshold function, by running AG on this g. For some linear threshold functions, D is exponential in n, and thus the approximation factor $O(\log D)$ is close to the trivial factor of n achieved by the naïve algorithm. Deshpande et al. proved that this bound cannot be improved by simplying constructing a different g, because for some linear threshold functions, the goal value of any suitable g is $2^{\Omega(n)}$.

Deshpande et al. achieved an approximation bound of $\alpha = 3$ using ADG in place of AG, with the above g. Showing that $\alpha = 3$ here is based on the following observations. (Here we use our simplifying assumption that all coefficients of the linear threshold formula are non-negative, but the bound still holds in the general case.) Setting b to $*^n$ in the ratio defining α, the numerator is the sum of the individual increases in utility that would be obtained by testing each variable in C alone. The denominator is $Q = Q_0 Q_1$. The set of performed tests C can be divided into the 1-tests where $x_i = 1$, and the 0-tests where $x_i = 0$. Assume without loss of generality that $f = 1$, so the last test is a 1-test. Thus the sum of the a_i's for the 1-tests in C, excluding the last one, is less than θ. The last 1-test can increase g_1 by at most Q_i. Thus the contribution of the 1-tests to the numerator, in the ratio defining α, is at most $2Q_1Q_0$. The sum of the a_i's for the 0-tests in C is less than Q_0, since f was not equal to 0. Thus the contribution of the 0-tests to the numerator is at most Q_0Q_1. So the numerator is at most $3Q_0Q_1$, and the ratio is at most 3. The analysis also holds for $b \neq *^n$, since b defines an induced linear threshold evaluation problem.

3 Evaluating one of m linear threshold classifiers

We apply the previous techniques to the problem of determining the value of any one of m given linear threshold classifiers. We prove the following theorem.

Theorem 1. *Given a set of m linear threshold classifiers $\{f^1, \ldots, f^m\}$ of the form $\sum_{i=1}^{n} a_i x_i \geq \theta$ where θ and each a_i are integer valued, there is a polynomial-time $(2m+1)$-approximation algorithm solving the problem of determining the value of at least one of these formulas on an initially unknown input $x \in \{0,1\}^n$, in our stochastic setting.*

Proof. Again, for simplicity we assume the a_i are non-negative, but the proof extends to the general case. For each of the m classifiers, we construct g_0 and g_1 for the associated linear threshold functions, as described above. We then combine these $2m$ utility functions using the disjunctive construction. This yields a submodular cover function g whose goal value is reached as soon as we have enough information to determine the value of one of the m linear threshold classifiers. We can solve this evaluation problem by using ADG to solve the SSSC problem for g. It remains to bound the approximation factor achieved, by upper bounding the value of α for ADG in this case.

Consider the ratio in the expression for α. As before, we take $b = *^n$ in this ratio; the analysis also holds for other b. For $j = 1, \ldots, m$, let g_1^j be the g_1 for the j^{th} linear threshold formula in the ensemble, and let g_0^j be defined analogously. Let Q_1^j and Q_0^j be the associated goal values.

We divide the tests in C into 0-tests and 1-tests. Recall the k-partite hypergraph interpretation of g. Let a_i^j denote the coefficient of x_i in the j^{th} linear threshold formula. Consider a test x_i in C, other than the last test. Suppose we performed this test first, before any other tests. If x_i were a 1-test, we claim it would contribute at most $\sum_{j=1}^{m}(a_i^j)Q_0^j \prod_{k\neq j} Q_0^k Q_1^k$ to g. This is because the test alone cannot achieve the goal value for any g_i^j (since it wasn't the last test in C), and thus a_i^j vertices would be deleted from the part of the hypergraph for each g_1^j as a result of the test, and each of those vertices is contained in $\sum_{j=1}^{m} Q_0^j \prod_{k\neq j} Q_0^k Q_1^k$ hyperedges of the original hypergraph. Similarly, if x_i were a 0-test, it would contribute at most $\sum_{j=1}^{m}(a_i^j)Q_1^j \prod_{k\neq j} Q_0^k Q_1^k$ to g if performed first. Using the fact that before the last test is performed, no g_1^j has reached its goal value, for each g_1^j, the sum of $\sum_{j=1}^{m}(a_i^j)Q_0^j \prod_{k\neq j} Q_0^k Q_1^k$ over all tests x_i in C but the last is at most $2m\prod_k Q_0^k Q_1^k$. Thus in the expression for α, the total contribution to the numerator of all tests but the last is at most $2m\prod_k Q_0^k Q_1^k$. The last test on its own, can contribute at most $\prod_k Q_0^k Q_1^k$ to g. Thus the numerator in the expression for α is at most $(2m+1)\prod_k Q_0^k Q_1^k$. Since the denominator is the goal value of g, which is $\prod_k Q_0^k Q_1^k$, it follows that α is at most $2m+1$. □

4 Finding a value predicted by at least z of m classifiers

Let $\mathcal{F} = \{f^1, \ldots, f^m\}$ be an ensemble of linear threshold classifiers. Let $z \leq \lceil \frac{m}{2} \rceil$. Suppose we now want to continue testing until we find at least z classifiers that predict the same value. Because $z \leq \lceil \frac{m}{2} \rceil$, a set of z such classifiers must always exist.

As in the previous section, for each linear threshold classifier f^j we have the utility function g^j constructed from g_0^j and g_1^j. Recall that g_1^j reaches its goal value Q_1^j on b iff b contains a 1-certificate for f^j, and g_0^j reaches Q_0^j iff b contains a 0-certificate for f^j. We will now create a utility function g_1 with goal value Q_1 such that $g_1(x) = Q_1$ iff at least z of the m classifiers evaluate to 1 on input x. We will also define a corresponding g_0 with goal value Q_0 such that $g_0(x) = Q_0$ iff at least z of the m classifiers evaluate to 0 on input x. We then use the conjunctive construction to produce our final g. Clearly $g(b)$ reaches its goal value iff b has enough information to determine a value 0 or 1 such that at least z of the f^j predict that value.

To construct g_1, consider the monotone Boolean CNF formula over m variables $y_1, \ldots, y_m$, consisting of all $\binom{m}{z-1}$ clauses (disjunctions) containing exactly $m - z + 1$ of the variables. The value of this formula is 1 iff at least z of the y_i are set to 1. Replacing each y_i by g_1^i, we then use this formula to combine the g_i into the single utility function g_1. Specifically, for each clause, we apply the disjunctive construction to the g_1^i in that clause, yielding a new utility function corresponding to that clause. We then apply the conjunctive construction to the utility functions corresponding to the clauses to get g_1. We now calculate the goal value Q_1 of g_1. Let R denote the maximum goal value of any of the g_0^i and g_1^i. Since each clause of the CNF formula has $m - z + 1$ variables, the utility function for each clause has a goal value of at most R^{m-z+1}. Since there are $\binom{m}{z-1}$ clauses, the goal value Q_1 for g_1 is at most $\binom{m}{z-1}R^{m-z+1}$. It follows from the properties of the conjunctive and disjunctive constructions that since the CNF formula is satisfied iff at least z of its input variables is 1, g_1 reaches its goal value on $b \in \{0, 1, *\}^n$ iff at least z of the g_1^i reach their goal values on b, which holds iff b contains a 1-certificate for at least z of the linear threshold formulas f^i.

The same CNF formula can be used to produce g_0 from the g_0^i, with the same goal value as g_1. Finally, g is formed by applying the conjunctive construction to g_0 and g_1, so the goal value Q for g is at most $\binom{m}{z-1}^2 R^{2(m-z+1)}$.

We can now solve the problem of finding a value shared by at least z agreeing classifiers by solving the SSSC problem for g. ADG does not yield a good bound in this case, so we use AG. The approximation factor achieved by AG is $\ln(Q+1)$, which in this case is $O(\ln\binom{m}{z-1} + (m-z+1)\ln R)$. We thus have the following theorem.

Theorem 2. *Let $1 \leq z \leq \lceil \frac{m}{2} \rceil$. There is an approximation algorithm for finding a common value predicted by at least z of the m classifiers in an ensemble of m linear threshold formulas, in our stochastic setting. The algorithm achieves an approximation factor that is $O(\ln \binom{m}{z-1} + (m-z+1)\ln(D_{max}))$, where D_{max} is the maximum $\sum_{i=1}^{n} |a_i|$ over all m. The runtime of the algorithm is polynomial in n and $\binom{m}{z-1}$.*

In particular, for the problem of finding a majority value of m linear threshold classifiers, the algorithm achieves an approximation factor of $O(m \ln(mD_{max}))$.

The dependence of the runtime on $\binom{m}{z-1}$ comes from the fact that to run AG, we need to evaluate the constructed g on inputs b, and the construction of g gives rise to an arithmetic formula that is the size of the CNF used in the construction. That CNF has $\binom{m}{z-1}$ clauses.

We note that if we want to do a weighted majority of the classifiers, rather than an unweighted majority, we can construct a CNF formula computing the weighted majority formula (in time polynomial in 2^m), and then proceed as above to get a similar result as in the unweighted case.

The above theorems raise some immediate questions. We focus on the (unweighted) majority case. One question is whether using a different (AND,OR) circuit for majority in the above construction, rather than the given CNF, would yield a smaller value for Q. It can be shown that this is not the case, and the given CNF is essentially optimal (note that it is the OR gates that blow up the value of Q, because the disjunctive construction multiplies goal values).

Another natural question is whether we can reduce the time complexity by reducing the time needed to compute g in in the greedy step of AG. One idea is to use a different construction of g, corresponding to computing the majority function using dynamic programming. For example, we can define $g^{z,m}(b)$ to be the value of a utility function for finding z of m functions with agreeing values, and express g^z recursively by an (AND,OR) formula on the functions $g^{z,m-1}$, $g^{z-1,m-1}$ and g_1^m. Unfortunately, the resulting utility function has a goal value that is as much as an $\frac{m}{2}^{\text{th}}$ order factorial.

Acknowledgments

Sarah R. Allen was partially supported by an NSF Graduate Research Fellowship under Grant 0946825 and by NSF grant CCF-1116594. Lisa Hellerstein was partially supported by NSF Grants 1217968 and 0917153.

References

[1] Sarah R. Allen. Approximately optimal testing strategies for systems of boolean functions. Master's thesis, Polytechnic Institute of New York University, 2013.

[2] Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, Olivier Chapelle, and Dor Kedem. Classifier cascade for minimizing feature evaluation cost. *Journal of Machine Learning Research - Proceedings Track*, 22:218–226, 2012.

[3] ICML workshop on learning with test-time budgets. `https://sites.google.com/site/budgetedlearning2013/`.

[4] Haim Kaplan, Eyal Kushilevitz, and Yishay Mansour. Learning with attribute costs. *STOC*, pages 356–365, 2005.

[5] Tonguç Ünlüyurt. Sequential testing of complex systems: a review. *Discrete Applied Mathematics*, 142(1-3):189–205, 2004.

[6] Louis Anthony Cox, Yuping Qiu, and Warren Kuehner. Heuristic least-cost computation of discrete classification functions with uncertain argument values. *Annals of Operations Research*, 21:1–29, 1989.

[7] Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. Approximation Algorithms for Stochastic Boolean Function Evaluation and Stochastic Submodular Set Cover. *CoRR*, /abs/1303.0726, 2013.

[8] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res. (JAIR)*, 42:427–486, 2011.

[9] Andrew Guillory and Jeff Bilmes. Simultaneous learning and covering with adversarial noise. *ICML*, 2011.

[10] Gowtham Bellala, Suresh K. Bhavnani, and Clayton Scott. Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Trans. on Information Theory*, 2012.

[11] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. *NIPS*, pages 766–774, 2010.