Chapter 23
Active Learning in Recommender Systems

Neil Rubens, Dain Kaplan, and Masashi Sugiyama

23.1 Introduction

Recommender Systems (RSs) are often assumed to present items to users for one reason – to *recommend* items a user will likely be interested in. Of course RSs *do* recommend, but this assumption is biased, with no help of the title, towards the "recommending" the system will do. There is another reason for presenting an item to the user: to learn more about his/her preferences, or his/her *likes* and *dislikes*. This is where Active Learning (AL) comes in. Augmenting RSs with AL helps the user become more self-aware of their own likes/dislikes while at the same time providing new information to the system that it can analyze for subsequent recommendations. In essence, applying AL to RSs allows for personalization of the recommending process, a concept that makes sense as recommending is inherently geared towards personalization. This is accomplished by letting the system actively influence which items the user is exposed to (e.g. the items displayed to the user during sign-up or during regular use), and letting the user explore his/her interests freely.

Unfortunately, there are very few opportunities for the system to acquire information, such as when a user rates/reviews an item, or through a user's browsing history.[1] Since these opportunities are few, we want to be as sure as possible that the data we acquire tells us something *important* about the user's preferences. After all, one of the most valuable assets of a company is user data.

Neil Rubens
University of Electro-Communications, Tokyo, Japan, e-mail: rubens@hrstc.org

Dain Kaplan
Tokyo Institute of Technology, Tokyo, Japan e-mail: dain@cl.cs.titech.ac.jp

Masashi Sugiyama
Tokyo Institute of Technology, Tokyo, Japan e-mail: sugi@cs.titech.ac.jp

[1] There is an increasing trend to utilize social networks for acquiring additional data (see Chapters 18 and 19).

F. Ricci et al. (eds.), *Recommender Systems Handbook*,
DOI 10.1007/978-0-387-85820-3_23,

For example, when a new user starts using a recommender system, very little is known about his/her preferences [44, 36, 2]. A common approach to learning the user's preferences is to ask him/her to rate a number of items (known as training points). A model that approximates the user's preferences is then constructed from this data. Since the number of items reviewed by the user cannot span the system's entire catalog (and indeed would make the task of AL as well as *recommending* moot points), the collection of items presented to the user for review must necessarily be very limited. The accuracy of the learned model thus greatly depends on the selection of good training points. A system might ask the user to rate *Star Wars I*, *II*, and *III*. By rating all three volumes of this trilogy, we will have a good idea of the user's preferences for *Star Wars*, and possibly by extension, an inclination for other movies within the Sci-Fi genre, but overall the collected knowledge will be limited. It is therefore unlikely that picking the three volumes of a trilogy will be informative.[2] Another issue with selecting a popular item such as *Star Wars* is that by definition the majority of people like them (or they would not be popular). It is not surprising then, that often little insight is gained by selecting popular items to learn about the user (unless the user's tastes are atypical).

There is sometimes a notion that AL is a bothersome, intrusive process, but it does not have to be this way [54, 38]. If the items presented to the user are interesting, it could be both a process of discovery and of exploration. Some Recommender Systems provide a "surprise me!" button to motivate the user into this explorative process, and indeed there are users who browse suggestions just to see what there is without any intention of buying. Exploration is crucial for users to become more self-aware of their own preferences (changing or not) and at the same time inform the system of what they are. Keep in mind that in a sense users can also be defined by the items they consume, *not* only by the ratings of their items, so by prompting users to rate different items it may be possible to further distinguish their preferences from one another and enable the system to provide better personalization and to better suit their needs.

This chapter is only a brief foray into Active Learning in Recommender Systems.[3] We hope that this chapter can, however, provide the necessary foundations.

For further reading, [46] gives a good, general overview of AL in the context of Machine Learning (with a focus on Natural Language Processing and Bioinformatics). For a theoretical perspective related to AL (a major focus in the field of Experimental Design), see [7, 4, 22]; there have also been recent works in Computer Science [16, 5, 51].

[2] Unless our goal is to learn a kind of micro-preference, which we can define as a person's tendency to be more 'picky' concerning alternatives close to one another in a genre they like.

[3] Supplementary materials on Active Learning can be found at: http://www.DataMilking.org/AL

23.1.1 Objectives of Active Learning in Recommender Systems

Different RSs have different objectives (see Chapter 8), which necessitate different objectives for their Active Learning components as well. As a result, one AL method may be better suited than another for satisfying a given task [35]. For example, what is important in the recommender system being built (see Chapter 10)? The difficulty of signing-up (user effort)? If the user is happy with the service (user satisfaction)? How well the system can predict a user's preferences (accuracy)? How well the system can express a user's preferences (user utility)? How well the system can serve other users by what it learns from this one (system utility)? System functionality may also be important, such as when a user inquires about a rating for an item of interest the system has insufficient data to predict a rating for, what the system does in response. Does it in such a case give an ambiguous answer, allowing the user to train the system further if they have the interest and the time to do so? Or does it require them to rate several other items before providing a prediction? Perhaps the user has experienced the item (e.g. watched the movie or trailer) and thinks their rating differs substantially from the predicted one [10]. In all these cases how the system responds to the user is important for consideration.

Traditionally AL does not consider the trade-off of exploration (learning user's preferences) and exploitation (utilizing user's preferences), that is, it does not dynamically assign weights to exploitation/exploration depending on system objectives. This trade-off is important because for a new user about which nothing or little is known, it may be beneficial to validate the worth of the system by providing predictions the user is likely to be interested in (exploitation), while long-term users may wish to expand their interests through exploration [38, 41].

Though an objective of the RS will likely be to provide accurate predictions to the user, the system may also need to recommend items of high novelty/serendipity, improve coverage, maximize profitability, or determine if the user is even able to evaluate a given item, to name a few [44, 21, 33]. Multiple objectives may need to be considered simultaneously (see Chapter 24), e.g. minimizing the net acquisition cost of training data *while* maximizing net profit, or finding the best match between the cost of offering an item to the user, the utility associated with expected output, and the alternative utility of inaction [38]. The utility of training may also be important, e.g. predicting ratings for exotic cars may not be so useful if the user is not capable of purchasing them and so should be avoided. It can be seen that the system objective is often much more complex than mere predictive accuracy, and may include the combination of several objectives.

While Recommender Systems in general often have an ill-defined or open-ended objective, namely to predict items a user would be interested in, Conversation-based AL [32, 37, 9], as the name suggests, engages in a conversation with the user as a goal oriented approach. It seeks to, through each iteration of questioning, elicit a response from the user to best reduce the search space for quickly finding what it is the user seeks (Section 23.8).

The New User Problem When a user starts using a RS they expect to see interesting results after a minimal amount of training. Though the system knows little about the user's preferences, it is essential that training points are selected to be rated by the user that will maximize the understanding of what the new user wants [35].

The New Product Problem As new products are introduced into the system, it is important to quickly improve prediction accuracy for these items by selecting users to rate them [24].

Cost of obtaining an output value Different means of obtaining an output value come at different costs. Implicit strategies, such as treating a user click on a suggested item as positive output, or not clicking as negative, are inexpensive in relation to user effort. Conversely, asking the user to explicitly rate an item is more costly, though still dependent on the task. Watching a movie like Star Wars to rate may provide good results but requires substantial user effort [20]; rating a joke requires much less. This often dovetails the exploration/exploitation coupling and trade-offs between obtaining outputs from different inputs should also be considered (e.g. certainty/uncertainty, ease of evaluation, etc.)

Adaptation for different AL methods Though we focus on the traditional objective of reducing predictive error, it is equally plausible to construct a method for maximizing other goals, such as profitability. In this case a model would pick points that most likely increase profit rather than a rating's accuracy.

23.1.2 An Illustrative Example

Let us look at a concrete example of Active Learning in a Recommender System. This is only meant to demonstrate concepts, so it is oversimplified. Please note that the similarity metric may differ depending on the method used; here, movies are assumed to be close to one another if they belong to the same genre. Figure 23.1 shows two charts, the leftmost is our starting state, in which we have already asked the user to rate a movie within the upper right group, which we will say is the Sci-Fi genre. The right chart shows us four possibilities for selecting our next training point: (a), (b), (c), or (d). If we select the training point (a) which is an obscure movie (like *The Goldfish Hunter*), it does not affect our predictions because no other movies (points) are nearby. If we select the training point (b), we can predict the values for the points in the same area, but these predictions are already possible from the training point in the same area (refer to the chart on the left). If training

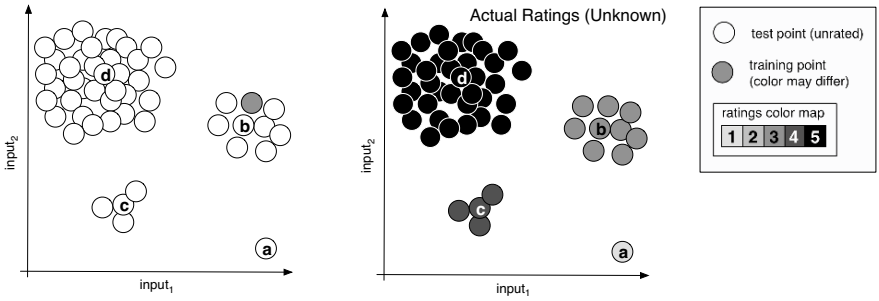


Fig. 23.1: Active Learning: illustrative example (See Section 23.1.2).

point (c) is selected, we are able to make new predictions, but only for the other three points in this area, which happens to be Zombie movies. By selecting training point (d), we are able to make predictions for a large number of test points that are in the same area, which belong to Comedy movies. Thus, selecting (d) is the ideal choice because it allows us to improve accuracy of predictions the most (for the highest number of training points).[4]

23.1.3 Types of Active Learning

AL methods presented in this chapter have been categorized based on our interpretation of their primary motivation/goal. It is important to note, however, that various ways of classification may exist for a given method, e.g. sampling close to a decision boundary may be considered as Output Uncertainty-based since the outputs are unknown, Parameter-based because the point will alter the model, or even Decision boundary-based because the boundary lines will shift as a result. However, since the sampling is performed with regard to decision boundaries, we would consider this the primary motivation of this method and classify it as such.

In addition to our categorization by primary motivation (Section 23.1), we further subclassify a method's algorithms into two commonly classified types for easier comprehension: instance-based and model-based.

Instance-based Methods A method of this type selects points based on their properties in an attempt to predict the user's ratings by finding the closest match to other users in the system, without explicit knowledge of the underlying model. Other common names for this type include memory-based, lazy learning, case-based, and non-parametric [2]. We assume that any existing data is accessible, as well as rating predictions from the underlying model.

[4] This may be dependent on the specific prediction method used in the RS.

Model-based Methods A method of this type selects points in an attempt to best construct a model that explains data supplied by the user to predict user ratings [2]. These points are also selected to maximize the reduction of expected error of the model. We assume that in addition to any data available to instance-based methods, the model and its parameters are also available.

Modes of Active Learning: Batch and Sequential Because users typically want to see the system output something interesting immediately, a common approach is to recompute a user's predicted ratings after they have rated a single item, in a sequential manner. It is also possible, however, to allow a user to rate several items, or several features of an item before readjusting the model. On the other hand, selecting training points sequentially has the advantage of allowing the system to react to the data provided by users and make necessary adjustments immediately. Though this comes at the cost of interaction with the user at each step. Thus a trade-off exists between Batch and Sequential AL: the usefulness of the data vs. the number of interactions with the user.

23.2 Properties of Data Points

When considering any Active Learning method, the following three factors should always be considered in order to maximize the effectiveness of a given point. Supplementary explanations are then given below for the first two. Examples refer to the Illustrative Example (Figure 23.1).

(R1) *Represented*: Is it already represented by the existing training set? E.g. point (b).

(R2) *Representative*: Is the point a good candidate for representing other data points? Or is it an outlier? E.g. point (a).

(R3) *Results*: Will selecting this point result in better prediction ratings or accomplish another objective? E.g. point (d), or even point (c).

(R1) Represented by the Training Data As explained in the introduction to this chapter, asking for ratings of multiple volumes from a trilogy, such as *Star Wars,* is not likely beneficial, as it may not substantially contribute to the acquisition of *new* information about the user's preferences. To avoid obtaining redundant information, therefore, an active learning method should favor items that are not yet well represented by the training set [18].

(R2) Representative of the Test Data It is important that any item selected for being rated by an AL algorithm be as representative of the test items as possible (we consider all items as potentially belonging to the test set), since the accuracy of the

algorithm will be evaluated based on these items. If a movie is selected from a small genre, like Zombie movies from the Illustrative Example (Figure 23.1), then obtaining a rating for this movie likely provides little insight into a user's preferences for other, more prominent genres. In addition, users naturally tend to rate movies from genres they like, meaning that any genre that dominates the training set (which is likely composed of items the user likes) may be representative of only a small portion of all items [38]. In order to increase information obtained, it is important to select representative items which may provide information about the other yet unrated items [18, 47, 53].

23.2.1 Other Considerations

In addition to the three Rs listed in Section 23.2, it may also be desirable to consider other criteria for data points, such as the following.

Cost As touched upon in the introduction to this chapter, obtaining implicit feedback from user selections is cheaper than asking the user to explicitly rate an item [19]. This can be considered a variable cost problem. One approach for tackling this, is to take into account both the cost of labeling an item and the future cost of estimated misclassification were the item to be added to the training set [27]. Moreover, the cost may be unknown beforehand [48].

Ratability A user may not always be able to provide a rating for an item; you cannot properly rate a movie you have not seen! It is suggested therefore that the probability of a user being able to evaluate an item also be considered [20].

Saliency Decision-centric AL places emphasis on items whose ratings are more likely to affect decision-making, and acquires instances that are related to decisions for which a relatively small change in their estimation can change the order of top rated predictions [43]. For example, unless labeling an item would result in displacing or rearranging a list of the top ten recommended movies on a user's home page (the salient items), it may be considered of little use. It is also possible to only consider the effect of obtaining an item's rating on items that are strongly recommended by the system [6].

Popularity It has also been suggested to take into account an item's popularity [35], i.e. how many people have rated an item. This operates on the principle that since a popular item is rated by many people, it may be rather informative. Conversely, an item's rating uncertainty should also be considered since positive items have a tendency to be rated highly by most users, indicating the item may not provide much discriminative power and thus not worth including in the training set.

Best/Worst It has been shown [29] that looking at the best/worst reviews is beneficial when a user makes a decision about an item. Extending this idea to Active Learning we hypothesize with the "best/worst" principle that in order to make a de-

cision about a user's preferences it may also be beneficial to obtain his/her best/worst ratings (as it may capture user preferences well). By asking a user to provide his/her most liked/disliked items, it changes the problem of AL to one in which a user is asked to provide a rating for an item in a known class (e.g. to select a favorite movie from within a liked genre), and the process of obtaining an item is what incurs the cost [30]. This process is called *active class selection.* This is opposite from traditional AL techniques in which the labeling process (and not the items themselves) is what is assumed to incur a cost.

23.3 Active Learning in Recommender Systems

With Traditional AL, users are asked to rate a set of preselected items. This is often at the time of enrollment, though a preselected list may be presented to existing users at a later date as well. It may be argued that since these items are selected by experts, they capture essential properties for determining a user's preferences. Conceptually this may sound promising, but in practice this often leads towards selecting items that best predict the preferences of only an *average* user. Since the idea of RS is to provide personalized recommendations, selecting items to rate in a personalized manner should readily make more sense.

23.3.1 Method Summary Matrix

The following matrix (Table 23.1) provides a summary of the methods overviewed in this chapter. Explicit performance numbers are not supplied because to our knowledge no such comprehensive comparison in fact exists. AL methods *could* be compared on an individual basis, but any results would be inconclusive. This is because authors have a tendency to fix the predictive method and *then* apply one or more compatible AL methods to compare performance. Moreover, AL methods are often designed for a specific predictive method, and may therefore not have good performance when applied to a different method (which creates potentially misleading results), or may not even be applicable if the underlying system is not able to provide the required information, e.g. distribution of rating estimates. For these reasons we have opted to omit performances figures of any kind.

23.4 Active Learning Formulation

Passive Learning (see Figure 23.2) refers to when training data is provided beforehand, or when the system makes no effort to acquire new data (it simply accumulates through user activities over time). *Active Learning*, on the other hand, selects train-

Primary Motivation of Approach	Description/Goal	Possible Considerations
Uncertainty Reduction (Section 23.5)	Reducing uncertainty of: • rating estimates (Section 23.5.1), • decision boundaries (Section 23.5.2), • model parameters (Section 23.5.3).	Reducing uncertainty may not always improve accuracy; the model could simply be certain about the wrong thing (e.g. when the predictive method is wrong).
Error Reduction (Section 23.6)	Reducing the predictive error by utilizing the relation between the error and: • the changes in the output estimates (Section 23.6.1.1), • the test set error (Section 23.6.1.2), • changes in parameter estimates (Section 23.6.2.1), • the variance of the parameter estimates (Section 23.6.2.2).	Estimating reduction of error reliably could be difficult and computationally expensive.
Ensemble-based (Section 23.7)	Identifying useful training points based on consensus between: • models in the ensemble (Section 23.7.1), • multiple candidate models (Section 23.7.1).	The effectiveness depends on the quality of models/candidates, and could be computationally expensive since it is performed with regards to multiple models/candidates.

Table 23.1: Method Summary Matrix.

ing points actively (the input) so as to observe the most informative output (user ratings, behavior, etc.).

Let us define the problem of active learning in a more formal manner. An item is considered to be a multi-dimensional input variable and is denoted by a vector $\mathbf{x}$ (also referred to as a *data point*).[5] The set of all items is denoted by $\mathcal{X}$. The preferences of a user u are denoted by a function f_u (also referred to as a *target function*); for brevity, we use f when referring to a target user. A rating of an item $\mathbf{x}$

[5] The way in which an item is represented depends on the RS and the underlying predictive method. In Collaborative Filtering based approaches items could be represented through the ratings of the users, or, in content based RSs, items could be represented through their descriptions.

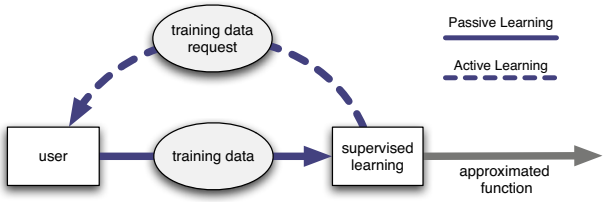


Fig. 23.2: Active learning employs an interactive/iterative process for obtaining training data, unlike passive learning, where the data is simply given.

is considered to be an output value (or *label*) and is denoted as $y = f(\mathbf{x})$. Each item $\mathbf{x}$ could be rated on a finite scale $\mathcal{Y} = \{1, 2, \ldots, 5\}$.

In supervised learning, the items and corresponding user ratings are often partitioned into complementary subsets – a training set and a testing set (also called a validation set). The task of supervised learning is then to, given a training set (often supplemented by the ratings of all users), learn a function $\widehat{f}$ that accurately approximates a user's preferences. Items that belong to the training set are denoted by $\mathcal{X}^{(Train)}$, and these items along with their corresponding ratings constitute a training set, i.e. $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{\mathbf{x}_i \in \mathcal{X}^{(Train)}}$. We measure how accurately the learned function predicts the true preferences of a user by the generalization error:

$$G(\widehat{f}) = \sum_{\mathbf{x} \in \mathcal{X}} \mathcal{L}\left(f(\mathbf{x}), \widehat{f}(\mathbf{x})\right) P(\mathbf{x}). \tag{23.1}$$

In practice, however, $f(\mathbf{x})$ is not available for all $\mathbf{x} \in \mathcal{X}$; it is therefore common to approximate the generalization error by the test error:

$$\widehat{G}(\widehat{f}) = \sum_{\mathbf{x} \in \mathcal{X}^{(Test)}} \mathcal{L}\left(f(\mathbf{x}), \widehat{f}(\mathbf{x})\right) P(\mathbf{x}), \tag{23.2}$$

where $\mathcal{X}^{(Test)}$ refers to the items in the *test set*, and prediction errors are measured by utilizing a loss function $\mathcal{L}$, e.g. mean absolute error (MAE):

$$\mathcal{L}_{MAE}\left(f(\mathbf{x}), \widehat{f}(\mathbf{x})\right) = \left|f(\mathbf{x}) - \widehat{f}(\mathbf{x})\right|, \tag{23.3}$$

or mean squared error (MSE):

$$\mathcal{L}_{MSE}\left(f(\mathbf{x}), \widehat{f}(\mathbf{x})\right) = \left(f(\mathbf{x}) - \widehat{f}(\mathbf{x})\right)^2. \tag{23.4}$$

The active learning criterion is defined so as to estimate the usefulness of obtaining a rating of an item $\mathbf{x}$ and adding it to the training set $\mathcal{X}^{(Train)}$ for achieving a certain objective (Section 23.1.1). For simplicity, let us consider this objective to be the minimization of generalization error of a learned fuction with respect to the training set. We then denote the active learning criterion as:

$$\widehat{G}(\mathcal{X}^{(Train)} \cup \{\mathbf{x}\}), \tag{23.5}$$

or for brevity, denote it as:

$$\widehat{G}(\mathbf{x}). \tag{23.6}$$

The goal of active learning is to select an item $\mathbf{x}$ that would allow us to minimize the generalization error $\widehat{G}(\mathbf{x})$:

$$\text{argmin}_{\mathbf{x}} \widehat{G}(\mathbf{x}). \tag{23.7}$$

If we consider asking a user to rate an item $\mathbf{x}_j$ or an item $\mathbf{x}_k$, then we would estimate their usefulness by an active learning criterion, i.e. $\widehat{G}(\mathbf{x}_j)$ and $\widehat{G}(\mathbf{x}_k)$, and select the one that will result in a smaller generalization error. Note that we need to estimate the usefulness of rating an item without knowing its actual rating. To distinguish a candidate item to be rated from the other items we refer to it as $\mathbf{x}_a$. AL can be applied to any predictive method as long as it provides the required information, such as rating estimates [42] and their distribution [23, 25], closeness to the decision boundary [55, 15], method parameters [49], etc.

$\mathbf{x}$	input (item)
$\mathcal{X}$	inputs (items)
y	output (item's rating)
$\mathcal{Y} = \{1, 2, \ldots, 5\}$	possible outputs (ratings), i.e. $y \in \mathcal{Y}$
f	user's preferences function (unknown to the system)
$\mathcal{X}^{(Train)}$	training inputs (rated items)
$\mathcal{T} = \{(\mathbf{x}_i, y_i\}_{\mathbf{x}_i \in \mathcal{X}^{(Train)}}$	training set (items and their ratings)
$\widehat{f}$	approximated function of user's preferences (from training set)
G	generalization error (predictive accuracy); see (23.1)
$\mathbf{x}_a$	item considered for rating
$\widehat{G}(\mathbf{x}_a)$	active learning criterion (estimates usefulness of rating an item $\mathbf{x}_a$)

Fig. 23.3: Summary of Notation.

Regression and Classification The problem of predicting a user's ratings could be treated as both a regression and a classification problem. It is a regression problem since the ratings are discrete numerical values, such as if we consider their ordinal properties, meaning the ratings could be ordered (e.g. a rating of 4 is higher than a rating of 3). On the other hand, we can disregard the numerical properties of the ratings and treat the problem as a classification one by treating ratings as classes/labels.[6] For example, we can use a nearest-neighbor (NN) approach to do classification, e.g. pick the most frequent label of the neighbors; or we can use NN to do regression, e.g. calculate the mean of the ratings of the neighbors. Throughout

[6] If the ordinal properties of the labels are considered, it is referred to as Ordinal Classification.

the chapter we use both classification and regression in examples, selecting the one most appropriate for aiding the current explanation.

23.5 Uncertainty-based Active Learning

Uncertainty-based AL tries to obtain training points so as to reduce uncertainty in some aspect, such as concerning output values [28], the model's parameters [23], a decision boundary [45], etc. A possible drawback to this approach is that reducing uncertainty may not always be effective. If a system becomes certain about user ratings, it does not necessarily mean that it will be accurate, since it could simply be certain about the wrong thing (i.e., if the algorithm is wrong, reducing uncertainty will not help). As an example, if the user has so far rated items positively, a system may mistakenly be certain that a user likes all of the items, which is likely incorrect.

23.5.1 Output Uncertainty

In Output Uncertainty-based methods, an item to label (training point) is selected so as to reduce the uncertainty of rating predictions for test items. In Figure 23.1, with the assumption that the RS estimates the rating of an item based on the cluster to which it belongs (e.g. items in the same movie genre receive the same rating), if a user's rating for a movie from the Sci-Fi genre (upper-right) has already been obtained, then there is a higher likelihood that the RS may be more certain about the ratings of other movies in the Sci-Fi genre, likely making it more beneficial to obtain a user's preference for a movie from a genre (cluster) not yet sampled, i.e. a cluster that is still uncertain.

The difference between instance-based and model-based approaches for Output Uncertainty-based AL is primarily in how, for an arbitrary item $\mathbf{x}$, the rating's distribution $P(Y_{\mathbf{x}})$ is obtained, where a rating's distribution is defined as the probability of an item being assigned a certain rating. For model-based methods it is possible to obtain the rating's distribution from the model itself. Probabilistic models are particularly well suited for this as they directly provide the rating's distribution [23, 25]. For instance-based methods, collected data is used to obtain the rating's distribution. As an example, methods utilizing nearest-neighbor techniques can obtain a rating's distribution based on the votes of its neighbors, where "neighbor" here means a user with similar preferences,[7] using a formula such as:

$$P(Y_{\mathbf{x}} = y) = \frac{\sum_{nn \in NN_{\mathbf{x},y}} w_{nn}}{\sum_{nn \in NN_x} w_{nn}}, \tag{23.8}$$

[7] Defining a neighbor as a similar item is also feasible depending on the method.

where NN_x are neighbors that have rated an item x, and $NN_{\mathbf{x},y}$ are neighbors that have given an item x a rating of y, and w_{nn} is the weight of the neighbor (such as similarity).

23.5.1.1 Active Learning Methods

Some AL methods [28] estimate the usefulness of a potential training point in a *local* (greedy) manner by measuring the uncertainty of its output value:

$$\widehat{G}_{Uncertainty_{local}}(\mathbf{x}_a) = -Uncertainty(Y_a). \tag{23.9}$$

Since our goal is to minimize $\widehat{G}$, rating an item with *high* uncertainty is useful; it will eliminate the uncertainty about the rating of the chosen item. However, labeling an item whose rating is uncertain does not necessarily accomplish the goal of reducing the uncertainty of ratings for other items (e.g. labeling an outlier may only reduce rating uncertainty for a few other similar items, such as when selecting item (c) in the Zombie genre, or even none as in (d), shown in Figure 23.1.

We may thus consider reducing uncertainty in a *global* manner by selecting an item which may reduce the uncertainty about *other* unrated items. One approach [40] for doing this is to define criteria by measuring the uncertainty of ratings over all of the test items $\mathcal{X}^{(Test)}$ with respect to a potential training input item $\mathbf{x}_a$:

$$\widehat{G}_{Uncertainty}(\mathbf{x}_a) = \frac{1}{\left|\mathcal{X}^{(Test)}\right|} \sum_{\mathbf{x}\in\mathcal{X}^{(Test)}} \mathbb{E}_{\mathcal{T}^{(a)}}\left(Uncertainty(Y_{\mathbf{x}})\right), \tag{23.10}$$

where $\frac{1}{\left|\mathcal{X}^{(Test)}\right|}$ is a normalizing factor, and $\mathbb{E}_{\mathcal{T}^{(a)}}\left(Uncertainty(Y_{\mathbf{x}})\right)$ is the expected value of uncertainty with respect to adding an estimated rating y_a of a candidate item $\mathbf{x}_a$ to the training set $\mathcal{T}$; i.e. $\mathcal{T}^{(a)} = \mathcal{T} \cup (\mathbf{x}_a, y_a)$.

A possible drawback of this non-local approach is that while with the local approach it is only necessary to estimate the uncertainty of a single output value y_a, for the non-local approach uncertainty needs to be estimated for the output values of *all* the test points *with respect to* a potential training point $(\mathbf{x}_a, y_a)$; this may be difficult to estimate accurately and could be computationally expensive.

23.5.1.2 Uncertainty Measurement

Uncertainty of an item's rating (output value) is often measured by its variance, its entropy [28], or by its confidence interval [38]. Variance is maximized when ratings deviate the most from the mean rating, and entropy when all the ratings are equally likely.

Uncertainty of an output value could be calculated by using a definition of variance as follows:

$$Uncertainty(Y_a) = VAR(Y_a) = \sum_{y \in \mathcal{Y}} \left(y - \overline{Y_a}\right)^2 P(Y_a = y), \tag{23.11}$$

where $\overline{Y_a}$ is the mean rating of all users for an item $\mathbf{x}_a$ and $P(Y_a = y)$ is the probability of an items rating Y_a being equal to y, both being calculated based on either nearest-neighbors for instance-based, or obtained from the model for model-based approaches.

Uncertainty could also be measured by entropy as follows:

$$Uncertainty(Y_a) = ENT(Y_a) = -\sum_{y \in \mathcal{Y}} P(Y_a = y) \log P(Y_a = y). \tag{23.12}$$

In [47] a method is proposed for measuring the uncertainty of a rating based on the probability of the most likely rating:

$$Uncertainty(Y_a) = -P(Y_a = y^*), \tag{23.13}$$

where $y^* = \text{argmax}_y P(Y_a = y)$ is the most likely rating.

In [38] the confidence interval is used as a measure of uncertainty for selecting the training input point:

$$c = P(b_l(Y_a) < y_a < b_u(Y_a)), \tag{23.14}$$

where c is the confidence that the actual rating y_a will lie in the interval between the lower bound $b_l(Y_a)$ and the upper bound $b_u(Y_a)$. For example, it is possible for the system to be certain that an item will be assigned a rating between 3 and 5 with a probability $c = 90\%$. Many methods prefer items with a higher upper bound, indicating that an item may be rated highly (good for exploitation), and if the confidence interval is also wide then it may be good for exploration. In some cases where it is desirable to increase the number of items predicted to be more highly rated, it may be beneficial to use the expected change in the lower bound of the confidence interval for selecting an item [38], the higher the expected change the more desirable.

23.5.2 Decision Boundary Uncertainty

In Decision Boundary-based methods, training points are selected so as to improve decision boundaries. Often an existing decision boundary is assumed to be somewhat accurate, so points are sampled close to the decision boundary to further refine it (Figure 23.4). In a way this may also be considered Output Uncertainty-based, since the uncertainty of the points close to the decision boundary may be high. This method operates with the assumption that the decision boundary of the underlying learning method (e.g. Support Vector Machine) is easily accessible. A clear advantage of this method is that given a decision boundary, selecting training examples by their proximity to it is computationally inexpensive.

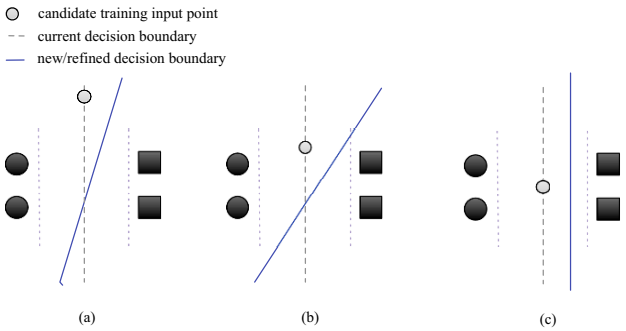


Fig. 23.4: Decision boundary uncertainty.

As discussed in [45], training points may be selected for obtaining a more accurate dividing hyperplane (Figure 23.4 (b)), or if the direction of the hyperplane is already certain, input points may be selected for reducing the size of margin (Figure 23.4 (c)). While it may seem obvious to sample training points closest to the decision boundary [55, 15], there are also methods that select the items furthest away [15] that have potential advantages in scenarios involving several candidate classifiers, which are discussed in Section 23.7. This is because a classifier should be quite certain about any items far from a decision boundary, but if newly acquired training data reveals the classifier to be inaccurate, the classifier may not fit the user's preferences well, so it should be removed from the pool of candidate classifiers.

23.5.3 Model Uncertainty

Model Uncertainty-based methods select training points for the purpose of reducing uncertainty within the model, more specifically, to reduce uncertainty about the model's parameters. The assumption is that if we improve the accuracy of the model's parameters the accuracy of output values will improve as well. If we were to predict a user's preferences based on membership in different interest groups [23], i.e. a group of people with a similar interest, then training points may be selected so as to determine to which groups the user belongs (Section 23.5.3.1).

23.5.3.1 Probabilistic Models

Probabilistic models are best explained with an example. The aspect model [23], a probabilistic latent semantic model in which users are considered to be a mixture of multiple interests (called aspects) is a good choice for this. Each user $u \in U$ has a probabilistic membership in different interest groups $z \in Z$. Users in the same interest group are assumed to have the same rating patterns (e.g. two users of the same

aspect will rate a given movie the same), so users and items $\mathbf{x} \in \mathcal{X}$ are independent from each other given the latent class variable z. The probability of the user u assigning an item $\mathbf{x}$ the rating y can be computed as follows:

$$P(y|\mathbf{x},u) = \sum_{z \in Z} p(y|\mathbf{x},z)p(z|u). \tag{23.15}$$

The first term $p(y|\mathbf{x},z)$ is the likelihood of assigning an item $\mathbf{x}$ the rating y by users in class z (approximated by a Gaussian distribution in [23]). It does not depend on the target user and represents the group-specific model. The global-model consists of a collection of group-specific models. The second term $p(z|u)$ is the likelihood for the target user u to be in class z, referred to as a user personalization parameter (approximated by a multinomial distribution in [23]). The user model $\boldsymbol{\theta}_u$ consists of one or more user personalization parameters, i.e. $\boldsymbol{\theta}_u = \{\theta_{u_z} = p(z|u)\}_{z \in Z}$.

A traditional AL approach would be to measure the usefulness of the candidate training input point $\mathbf{x}_a$ based on how much it would allow for reduction of the uncertainty about the user model's parameters $\boldsymbol{\theta}_u$ (i.e. the uncertainty about to which interest group z the user u belongs):

$$\widehat{G}_{\theta Uncertainty}(\mathbf{x}_a) = Uncertainty(\boldsymbol{\theta}_u), \tag{23.16}$$

$$Uncertainty(\boldsymbol{\theta}_u) = -\left\langle \sum_{z \in Z} \theta_{u_z|\mathbf{x}_a,y} \log \theta_{u_z|\mathbf{x}_a,y} \right\rangle_{p(y|\mathbf{x}_a,\boldsymbol{\theta}_u)}, \tag{23.17}$$

where $\boldsymbol{\theta}_u$ denotes the currently estimated parameters of the user u and $\theta_{u_z|x,y}$ a parameter that is estimated using an additional training point $(\mathbf{x}_a, y)$. Since the goal of the above criterion is to reduce the uncertainty of which interest groups the target user belongs to, it favors training points that assign a user to a *single* interest group. This approach may not be effective for all models, such as with the aspect model, in which a user's preferences are better modeled by considering that a user belongs to *multiple* interest groups [23, 25].

Another potential drawback comes from the expected uncertainty being computed over the distribution $p(y|\mathbf{x},\boldsymbol{\theta}_u)$ by utilizing the currently estimated model $\boldsymbol{\theta}_u$. The currently estimated model could be far from the true model, particularly when the number of training points is small, but the number of parameters to be estimated is large. Therefore, performing AL based only on a single estimated model can be misleading [25]. Let us illustrate this by the following example shown in Figure 23.5. The four existing training points are indicated by solid line contours, test points by dashed ones. Based on these four training examples, the most likely decision boundary is the horizontal line (dashed), even though the true decision boundary is a vertical line (solid). If we select training input points based only on the estimated model, subsequent training points would likely be obtained from areas along the estimated boundary, which are ineffective in adjusting the estimated decision boundary (horizontal line) towards the correct decision boundary (vertical line). This example illustrates that performing AL for the currently estimated model

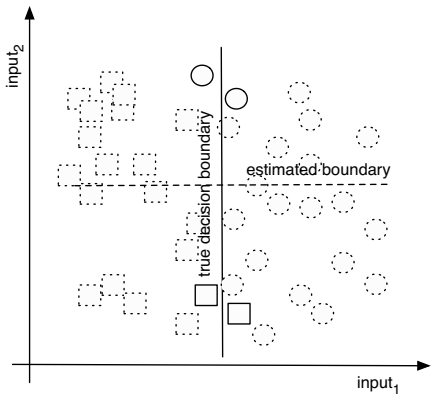


Fig. 23.5: A learning scenario when the estimated model is far from the true model. Training points are indicated by solid contours.

without taking into account the model's uncertainty can be very misleading, particularly when the estimated model is far from the true model. A better strategy could be to consider model uncertainty by utilizing the model distribution for selecting training input points [25]. This would allow for adjusting the decision boundary more effectively since decision boundaries other than the estimated one (i.e. horizontal line) would be considered for selecting the training input points. This idea is applied to probabilistic models in [25] as follows. The usefulness of the candidate training input point is measured based on how much it allows adjusting the model's parameters $\boldsymbol{\theta}_u$ towards the optimal model parameters $\boldsymbol{\theta}_u{}^*$:

$$\widehat{G}_{\theta Uncertainty}(\mathbf{x}_a) = \left\langle \sum_{z \in Z} \theta_{u_z}^* \log \frac{\theta_{u_z|\mathbf{x}_a,y}}{\theta_{u_z}^*} \right\rangle_{p(y|\mathbf{x}_a,\boldsymbol{\theta}_u{}^*)} . \tag{23.18}$$

The above equation corresponds to Kullback–Leibler divergence which is minimized when the estimated parameters are equal to the optimal parameters. The true model $\boldsymbol{\theta}_u{}^*$ is not known but could be estimated as the expectation over the posterior distribution of the user's model i.e. $p(\boldsymbol{\theta}_u|u)$.

23.6 Error-based Active Learning

Error-based Active Learning methods aim to reduce the predictive error, which is often the final goal. Instance-based approaches try to find and utilize the relation between the training input points and the predictive error. Model-based approaches tend to aim at reducing the model error (i.e. the error of model parameters), which is hoped would result in the improvement of predictive error.

23.6.1 Instance-based Methods

Instance-based methods aim at reducing error based on the properties of the input points, such as are listed in Section 23.2.

23.6.1.1 Output Estimates Change (Y-Change)

This approach [42] operates on the principle that if rating estimates do not change then they will not improve. Thus, if the estimates of output values do change, then their accuracy may either increase or decrease. However, it is expected that at least something will be learned from a new training point, so it follows then that in many cases estimates do in fact become more accurate. Assuming that most changes in estimates are for the better, an item that causes many estimates to change will result in the *improvement* of many estimates, and is considered useful.

As an example (Figure 23.6), if a user rates an item that is representative of a large genre, such as the Sci-Fi movie *Star Wars*, then its rating (regardless of its value) will likely cause a change in rating estimates for many other related items (e.g. items within that genre), in other words, rating such a representative item is very informative about the user's preferences. On the other hand, the user rating an item without many other similar items, such as the movie *The Goldfish Hunter*, would change few rating estimates, and supply little information.

To find the expected changes in rating estimates caused by a candidate item's rating, all possible item ratings are considered (since the true rating of a candidate item is not yet known). The difference is calculated between rating estimates for each item for each of its possible ratings, before and after it was added to the training set (refer to the pseudocode in Algorithm 1).

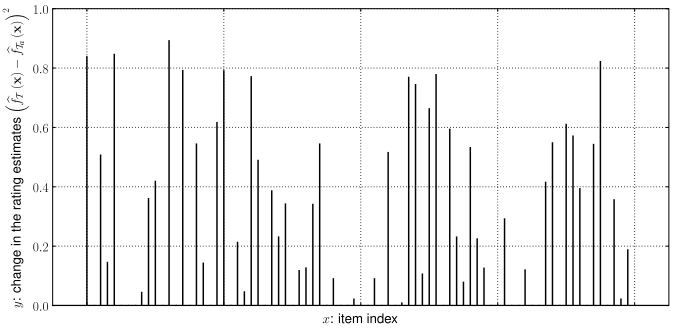
More formally the above criterion could be expressed as:

$$\widehat{G}_{Y\,change}(\mathbf{x}_a) = -\sum_{\mathbf{x}\in\mathcal{X}^{(Test)}} \mathbb{E}_{y\in\mathcal{Y}}\mathcal{L}(\widehat{f}_{\mathcal{T}}(\mathbf{x}), \widehat{f}_{\mathcal{T}\cup(\mathbf{x}_a,y)}(\mathbf{x})), \tag{23.19}$$

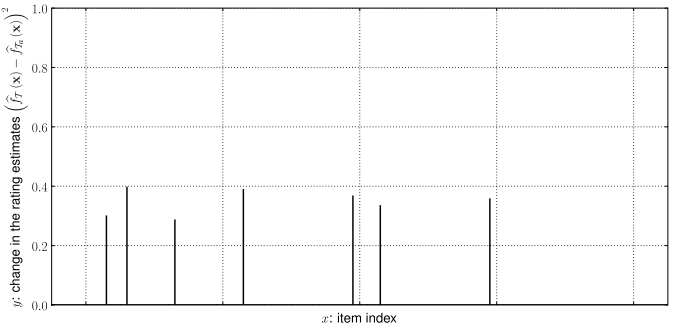
where $\widehat{f}_{\mathcal{T}}(\mathbf{x})$ is the estimated rating for an item $\mathbf{x}$ given the current training set $\mathcal{T}$, and $\widehat{f}_{\mathcal{T}\cup(\mathbf{x}_a,y)}(\mathbf{x})$ is the rating's estimate after a hypothetical rating y of an item $\mathbf{x}_a$ is added to the training set $\mathcal{T}$, and $\mathcal{L}$ is the loss function that measures the differences between the rating estimates $\widehat{f}_{\mathcal{T}}(\mathbf{x})$ and $\widehat{f}_{\mathcal{T}\cup(\mathbf{x}_a,y)}(\mathbf{x})$. By assuming that ratings of a candidate item are equally likely and using a mean squared loss function, the above criterion could be written as:

$$\widehat{G}_{Y\,change}(\mathbf{x}_a) = -\sum_{\mathbf{x}\in\mathcal{X}^{(Test)}} \frac{1}{|\mathcal{Y}|}\sum_{y\in\mathcal{Y}}\left(\widehat{f}_{\mathcal{T}}(\mathbf{x}) - \widehat{f}_{\mathcal{T}\cup(\mathbf{x}_a,y)}(\mathbf{x})\right)^2 \tag{23.20}$$

where $\frac{1}{|\mathcal{Y}|}$ is a normalizing constant since we assume all possible ratings $y \in \mathcal{Y}$ of an item $\mathbf{x}_a$.



(a)



(b)

Fig. 23.6: Output estimate-based AL (Section 23.6.1.1). The x-axis corresponds to an item's index, and the y-axis to the changes in rating estimates with regard to a candidate training point. Training points that cause many changes in rating estimates are considered to be more informative (a).

The advantage of this criterion is that it relies only on the *estimates* of ratings, available from any learning method. It has a further advantage of utilizing all unrated items, something that differentiates it from other methods in which only a small subset of all items (ones that have been rated by the user) are considered. It also works in tandem with any of a variety of learning methods, enabling it to potentially adapt to different tasks.

23.6.1.2 Cross Validation-based

In this approach a training input point is selected based on how well it may allow for approximation of already known ratings, i.e. items in the training set [15]. That is, a

Algorithm 1 Output estimates-based Active Learning (Section 23.6.1.1).

$\widehat{G}$ estimates predictive error that rating an item $\mathbf{x}_a$ would achieve
function $\widehat{G}(\mathbf{x}_a)$
  # learn a preference approximation function $\widehat{f}$ based on the current training set $\mathcal{T}$
  $\widehat{f}_{\mathcal{T}}$ =learn($\mathcal{T}$)
    # for each possible rating of an item $\mathbf{x}_a$ e.g. $\{1,2,\ldots,5\}$
    for $y_a \in \mathcal{Y}$
      # add a hypothetical training point $(\mathbf{x}_a, y_a)$
      $\mathcal{T}^{(a)} = \mathcal{T} \cup (\mathbf{x}_a, y_a)$
      # learn a new preference approximation function $\widehat{f}$ based on the new training set $\mathcal{T}^{(a)}$
      $\widehat{f}_{\mathcal{T}^{(a)}}$ =learn($\mathcal{T}^{(a)}$)
      # for each unrated item
      for $\mathbf{x} \in \mathcal{X}^{(Test)}$
        # record the differences between ratings estimates
        # before and after a hypothetical training point $(\mathbf{x}_a, y_a)$ was added to the training set $\mathcal{T}$
        $\widehat{G} = \widehat{G} + \left(- \left(\widehat{f}_{\mathcal{T}}(\mathbf{x}) - \widehat{f}_{\mathcal{T}^{(a)}}(\mathbf{x}) \right)^2 \right)$
  return $\widehat{G}$

candidate training point $\mathbf{x}_a$ with each possible rating $y \in \mathcal{Y}$ is added to the training set $\mathcal{T}$, then an approximation of the user's preferences $\widehat{f}$ is obtained and its accuracy is evaluated (i.e. cross-validated) on the training items $\mathcal{X}^{(Train)}$. It is assumed that when the candidate training item is paired with its correct rating, the cross-validated accuracy will improve the most. The usefulness of the candidate training point is measured by the improvement in the cross-validated accuracy as following:

$$\widehat{G}_{CV_{\mathcal{T}}}(\mathbf{x}_a) = -\max_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}^{(Train)}} \mathcal{L}(\widehat{f}_{\mathcal{T} \cup (\mathbf{x}_a, y)}(x), f(x)), \tag{23.21}$$

where $\mathcal{L}$ is a loss function such as MAE or MSE (Section 23.4), and $f(x)$ is the actual rating of the item $\mathbf{x}$, and $\widehat{f}_{\mathcal{T} \cup (\mathbf{x}_a, y)}(x)$ is the approximated rating (where a function $\widehat{f}$ is learned from the training set $\mathcal{T} \cup (\mathbf{x}_a, y)$) .

A potential drawback is that training points selected by this AL method could be overfitted to the training set.

23.6.2 Model-based

In model-based approaches training input points are obtained as to reduce the model's error, i.e. the error of the model's parameters. A potential drawback of this approach is that reducing the model's error may not necessarily reduce the prediction error which is the objective of AL.

23.6.2.1 Parameter Change-based

Parameter Change-based AL [49] favors items that are likely to influence the model the most. Assuming that changes in the model's parameters are for the better, i.e. approach the optimal parameters, it is then beneficial to select an item that has the greatest impact on the model's parameters:

$$\widehat{G}_{\theta change}(\mathbf{x}_a) = -\sum_{\theta} \mathbb{E}_{y \in \mathcal{Y}} \mathcal{L}(\theta_{\mathcal{T}}, \theta_{\mathcal{T} \cup (\mathbf{x}_a, y)}), \tag{23.22}$$

where $\theta_{\mathcal{T}}$ are the model's parameters estimated from the current training set $\mathcal{T}$, and $\theta_{\mathcal{T} \cup (\mathbf{x}_a, y)}$ are the model's parameter estimates after a hypothetical rating y of an item $\mathbf{x}_a$ is added to the training set $\mathcal{T}$, and $\mathcal{L}$ is the loss function that measures the differences between the parameters.

23.6.2.2 Variance-based

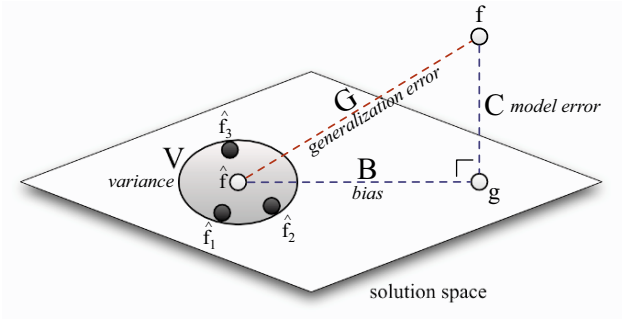


Fig. 23.7: Decomposition of generalization error G into model error C, bias B, and variance V, where g denotes optimal function, $\widehat{f}$ is a learned function $\widehat{f}_i$'s are the learned functions from a slightly different training set.

In this approach the error is decomposed into three components: model error C (the difference between the optimal function approximation g, given the current model, and the true function f), bias B (the difference between the current approximation $\widehat{f}$ and an optimal one g), and variance V (how much the function approximation $\widehat{f}$ varies). In other words, we have:

$$G = C + B + V. \tag{23.23}$$

One solution [13] is to minimize the variance component V of the error by assuming that the bias component becomes negligible (if this assumption is not satisfied then this method may not be effective). There are a number of methods proposed that

aim to select training inputs for reducing a certain measure of the variance of the model's parameters. The A-optimal design [11] seeks to select training input points so as to minimize the average variance of the parameter estimates, the D-optimal design [26] seeks to maximize the differential Shannon information content of the parameter estimates, and the Transductive Experimental design [56] seeks to find representative training points that may allow retaining most of the information of the test points. The AL method in [51], in addition to the variance component, also takes into account the existense of the model error component.

23.6.2.3 Image Restoration-based

It is also possible to treat the problem of predicting the user's preferences as one of image restoration [34], that is, based on our limited knowledge of a user's preferences (a partial picture), we try to restore the complete picture of the user's likes and dislikes. The AL task is then to select the training points that would best allow us to restore the "image" of the user's preferences. It is interesting to note that this approach satisfies the desired properties of the AL methods outlined in Section 23.2. For example, if a point already exists in a region, then without sampling neighboring points the image in that region could likely be restored. This approach also may favor sampling close to the edges of image components (decision boundaries).

23.7 Ensemble-based Active Learning

Sometimes instead of using a single model to predict a user's preferences, an ensemble of models may be beneficial (see Chapter 21). In other cases only a single model is used, but it is selected from a number of candidate models. The main advantage of this is the premise that different models are better suited to different users or different problems. The preferences of one user, for example, could be better modeled by a stereotype model, while the preferences of another user may be better modeled by a nearest-neighbor model. The training input points for these AL methods must be selected with regards to multiple models (Section 23.7.1) or multiple model candidates (Section 23.7.2).

23.7.1 Models-based

In Models-based approaches, the models form a "committee" of models that act, in a sense, cooperatively to select training input points [50]. Methods tend to differ with respect to: (1) how to construct a committee of models, and (2) how to select training points based on committee members [46]. As [46] explains thoroughly (please refer to it for more details), the Query by Committee approach (QBC) in-

volves maintaining a committee of models which are all trained on the same training data. In essence, they represent competing hypotheses for what the data might look like (as represented by the model). The members of this committee then vote on how to label potential input points (the "query" in "QBC"). The input points for which they disagree the most are considered to be the most informative. The fundamental premise of QBC is minimizing the version space, or the subset of all hypotheses that are consistent with all the collected training data; we want to then constrain the size of this space as much as possible, while at the same time minimizing the number of training input points. Put a different way, QBC "queries" in controversial regions to refine the version space.

There are many ways to construct the committee of models; [46] provides numerous examples. It can, for example, be constructed through simple sampling [50]. With generative model classes, this can be achieved by randomly sampling an arbitrary number of models from some posterior distribution, e.g. using the Dirichlet distribution over model parameters for naive Bayes [31], or sampling Hidden Markov Models (HMMs) using the Normal distribution [14]. The ensemble can be constructed for other model classes (such as discriminative or non-probabilistic models) as well, e.g. query-by-boosting and query-by-bagging [1], which employ the boosting [17] and bagging [8] ensemble learning methods to construct the committees; there has also been research [12] on using a selective sampling algorithm for neural networks that utilizes the combination of the "most specific" and "most general" models (selecting the models that lie at two extremes of the current version space given the current training set).

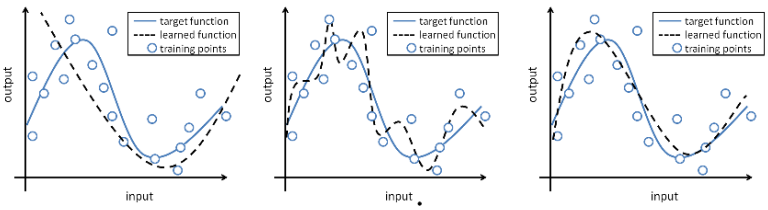
The "committee is still out" on the appropriate number of models to use, but even small sizes have demonstrated good results [50, 31, 47].

Measuring the disagreement between models is fundamental to the committee approach; there are two main means for calculating disagreement: vote uncertainty [14] and average Kullback-Leibler (KL) divergence [31]. Vote uncertainty selects the point with the largest disagreement between models of the committee. KL divergence is an information-theoretic measure of the difference between two probability distributions. KL divergence selects the input point with the largest average difference between the distributions of the committee consensus and the most differing model.

23.7.2 Candidates-based

Different models are better suited to different users or to different problems (see Chapter 2). So both the choice of the training set (AL) and the choice of the model, called Model Selection (MS), affect the predictive accuracy of the learned function. There is in fact a strong dependency between AL and MS, meaning that useful points for one model may not be as useful for another (Figure 23.9). This section discusses how to perform AL with regards to multiple model candidates and the issues that may arise when doing so.

The concept of *model* has several different meanings. We may refer to a model as a set of functions with some common characteristic, such as a function's complexity, or the type of a function or learning method (e.g. SVM, Naive Bayes, nearest-neighbor, or linear regression). The characteristics of the functions that may differ are often referred to as parameters. Thus, given a model and training data, the task of MS is to find parameters that may allow for accurate approximation of the target function. All of the model's characteristics affect the predictive accuracy, but for simplicity we concentrate only on the complexity of the model.



(a) under-fit **(b)** over-fit **(c)** appropriate fit

Fig. 23.8: Dependence between model complexity and accuracy.

As illustrated by Figure 23.8, if the model is too simple in comparison with the target function, then the learned function may not be capable of approximating the target function, making it under-fit (Figure 23.8a). On the other hand, if the model is too complex it may start trying to approximate irrelevant information (e.g. noise that may be contained in the output values) which will cause the learned function to over-fit the target function (Figure 23.8b). A possible solution to this is to have a number of candidate models. The goal of model selection (MS) is thus to determine the weights of the models in the ensemble, or in the case of a single model being used, to select an appropriate one (Figure 23.8c):

$$\min_{\mathcal{M}} G(\mathcal{M}). \tag{23.24}$$

The task of AL is likewise to minimize the predictive error, but with respect to the choice of the training input points:

$$\min_{\mathcal{X}^{(Train)}} G(\mathcal{X}^{(Train)}). \tag{23.25}$$

It would be beneficial to combine AL and MS since they share a common goal of minimizing the predictive error:

$$\min_{\mathcal{X}^{(Train)},\mathcal{M}} G(\mathcal{X}^{(Train)},\mathcal{M}). \tag{23.26}$$

Ideally we would like to choose the model of appropriate complexity by a MS method and to choose the most useful training data by an AL method. However

simply combining AL with MS in a batch manner, i.e. selecting all of the training points at once, may not be possible due to the following paradox:

- To select training input points by a standard AL method, a model must be fixed. In other words, MS has already been performed (see Figure 23.9).
- To select the model by a standard MS method, the training input points must be fixed and corresponding training output values must be gathered. In other words, AL has already been performed (see Figure 23.10).

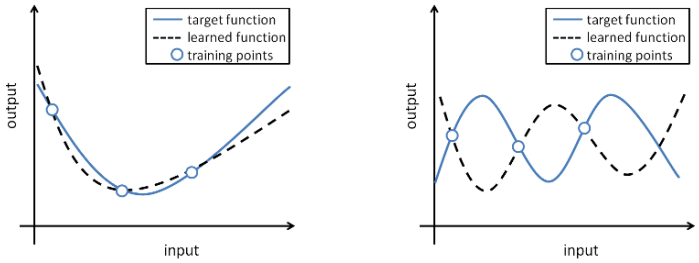
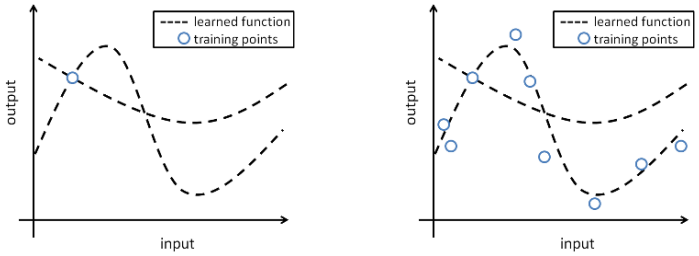


Fig. 23.9: Training input points that are good for learning one model, are not necessary good for the other.



Unable to determine which model is more appropriate (Model Selection), until training points have been obtained (Active Learning).

Fig. 23.10: Dependence of Model Selection on Active Learning.

As a result Batch AL selects training points for a randomly chosen model, but after the training points are obtained the model is selected once again, giving rise to the possibility that the training points will not be as useful if the initial and final models differ. This means that the training points could be over-fitted to a possibly inferior model, or likewise under-fitted.

With Sequential AL, the training points and models are selected incrementally in a process of selecting a model, then obtaining a training point for this model, and so on. Although this approach is intuitive, it may perform poorly due to *model*

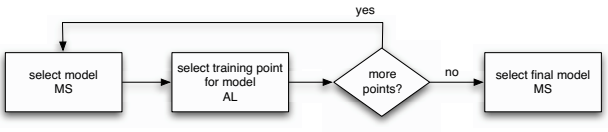


Fig. 23.11: Sequential Active Learning.

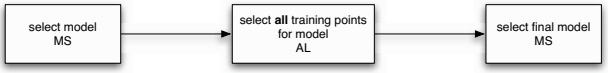


Fig. 23.12: Batch Active Learning.

drift, where a chosen model varies throughout the learning process. As the number of training points increases, more complex models tend to fit data better and are therefore selected over simpler models. Since the selection of training input points depends on the model, the training points chosen for a simpler model in the early stages could be less useful for the more complex model selected at the end of the learning process. Due to model drift, portions of training points are gathered for different models, resulting in the training data being not well suited for any of the models. However, because the selection of the final model is unclear at the onset, one possibility is to select training input points with respect to multiple models [52], by optimizing the training data for all the models:

$$\min_{\mathcal{X}^{(Train)}} \sum_{\mathcal{M}} \widehat{G}(\mathcal{X}^{(Train)}, \mathcal{M}) w(\mathcal{M}), \tag{23.27}$$

where $w(\mathcal{M})$ refers to the weight of the model in the ensemble, or among the candidates. This allows each model to contribute to the optimization of the training data and thus the risk of overfitting the training set to possibly inferior models can be hedged.

23.8 Conversation-based Active Learning

Differing from standard AL in which the goal is to obtain ratings for dissimilar items (for improving prediction accuracy over the entire set), Conversation-based AL is goal oriented with the task of starting general and, through a series of interaction cycles, narrowing down the user's interests until the desired item is obtained [32, 37, 9], such as selecting a hotel to stay at during a trip. In essence, the goal is to supply the user with the information that best enables them to reduce the set of possible items, finding the item with the most utility. The system therefore aims at making accurate predictions about items with the highest utility for a potentially small group of items, such as searching for a restaurant within a restricted locale.

A common approach is to iteratively present sets of alternative recommendations to the user, and by eliciting feedback, guide the user towards an end goal in which the scope of interest is reduced to a single item. This cycle-based approach can be beneficial since users rarely know all their preferences at the start (becoming self-aware), but tend to form and refine them during the decision making process (exploration). Thus Conversation-based AL should also allow users to refine their preferences in a style suitable to the given task. Such systems, unlike general RSs, also include AL by design, since a user's preferences are learned through active interaction. They are often evaluated by the predictive accuracy, and also by the length of interaction before arriving at the desired goal.

23.8.1 Case-based Critique

One means for performing a conversation with a user is the Case-based Critique approach, which finds cases similar to the user's query or profile and then elicits a critique for refining the user's interests (see Chapters [37] and 13). As mentioned above (Section 23.8), the user is not required to clearly define their preferences when the conversation initiates; this may be particularly beneficial for mobile device-oriented systems. Each step of iteration displays the system's recommendations in a ranked list and allows for user critique, which will force the system to re-evaluate its recommendations and generate a new ranked list. Eliciting a user critique when a feature of a recommended item is unsatisfactory may be more effective in obtaining the end goal than mere similarity-based query revision combined with recommendation by proposing. As an example of a user critique, he/she may comment "I want a less expensive hotel room" or "I like restaurants serving wine."

23.8.2 Diversity-based

While suggesting items to the user that are similar to the user query is important (Section 23.8.1), it may also be worthwhile to consider diversity among the set of proposed items [32]. This is because if the suggested items are too similar to each other, they may not be representative of the current search space. In essence, the recommended items should be as representative and diverse as possible, which should be possible without appreciably affecting their similarity to the user query.

It is particularly important to provide diverse choices while the user's preferences are in their embryonic stages. Once the user knows what it is they want, providing items that match as closely as possible may be pertinent, and the AL technique used should attempt to make this distinction, i.e. if the recommendation space is properly focused, reduce diversity, and if incorrect, increase it.

23.8.3 Query Editing-based

Another possibility is to allow a user to repeatedly edit and resubmit a search query until their desired item is found [9]. Since it is an iterative process, the object is to minimize the number of queries needed before the user finds the item of highest utility. A query's usefulness is estimated based on the likelihood of the user submitting a particular query, along with its satisfiability, accomplished by observing user actions and inferring any constraints on user preferences related to item utility and updating the user's model. As an example, a user may query for hotels that have air-conditioning and a golf course. The RS can determine this to be satisfiable, and further infer that though the user is likely to add a restraint for the hotel being located in the city-center, no hotels match such criteria, so the system preemptively notifies the user that such a condition is unsatisfiable to prevent wasted user effort. The RS may also infer that for a small increase in price there are hotels with a pool and spa and a restaurant. Knowing the user's preferences for having a pool (and not for other options), the system would only offer adding the pool option, since it may increase the user's satisfaction, and not the others since they may overwhelm the user and decrease overall satisfaction.

23.9 Computational Considerations

It is also important to consider the computational costs of AL algorithms. [40] have suggested a number of ways of reducing the computational requirements, summarized (with additions) below.

- Many AL select an item to be rated based on its expected effect on the learned function. This may require retraining with respect to each candidate training item, and so efficient incremental training is crucial. Typically this step-by-step manner has lower cost than starting over with a large set.
- New rating estimates may need to be obtained with respect to each candidate item. Likewise, this could be done in an incremental manner, since only the estimates that change would need to be obtained again.
- It is possible to incrementally update the estimated error only for items likely to be effected by the inclusion of a training point, which in practice is only nearby items or items without similar features. A common approach is to use inverted indices to group items with similar features for quick lookup.
- A candidate training item's expected usefulness can likely be estimated using a subset of all items.
- Poor candidates for training points can be partially pruned through a prefiltering step that removes poor candidate items based on some criteria, such as filtering books written in a language the user cannot read. A suboptimal AL method may be a good choice for this task.

23.10 Discussion

Though very brief, hopefully the collection of Active Learning methods presented in this chapter has demonstrated that AL is indeed not only beneficial but also desirable for inclusion in many systems, namely Recommender Systems. It can be seen that due to individual characteristics, the AL method selected, in many cases, relies heavily on the specific objectives (Section 23.1.1) that must be satisfied, either due to business constraints, preferred system behavior, user experience, or a combination of these (and possibly others). In addition to AL objectives, it is also prudent to evaluate the computational costs (Section 23.9) of any methods under consideration for use, and their trade-offs. Despite the success that many of the methods discussed have received, there is also something to be said for abstracting the problem, or finding solutions to other problems that though seemingly unrelated, may have strikingly similar solutions (e.g. Image Restoration (Section 23.6.2.3)). We have also touched upon conversation-based systems (Section 23.8) which differ from traditional RSs, but include the notion of AL by design. Depending on the task at hand, such as specific goal oriented assistants, this may also be a nice fit for a Recommender System.

Some issues related to AL have already been well studied in Statistics; this is not the case in Computer Science, where research is still wanting. Recommender Systems are changing at a rapid pace and becoming more and more complex. An example of this is the system that won the NetFlix Recommendation Challenge, which combined multiple predictive methods in an ensemble manner (see Chapter 5). Given the high rate of change in predictive methods of RSs, and their complex interaction with AL, there is an ever increasing need for new approaches.

Improving accuracy has traditionally been the main focus of research. Accuracy alone, however, may not be enough to entice the user with RSs. This is because the system implementing AL may also need to recommend items of high novelty/serendipity, improve coverage, or maximize profitability, to name a few [44, 21, 33]. Another aspect that is frequently overlooked by AL researchers is the manner in which a user can interact with AL to reap improvements in performance. Simply presenting items to the user for rating lacks ingenuity to say the least; surely there is a better way? One example of this is a work [3] which demonstrated that by using the right interface even such menial tasks as labeling images could be made fun and exciting. With the right interface alone the utility of an AL system may increase dramatically.

Many issues remain that must be tackled to ensure the longevity of AL in RSs; with a little innovation and elbow grease we hope to see it transform from a "bothersome process" to an enjoyable one of self-discovery and exploration, satisfying both the system objectives and the user at the same time.

Acknowledgments

We would like to express our appreciation to Professor Okamoto, Professor Ueno, Professor Tokunaga, Professor Tomioka, Dr. Sheinman, Dr. Vilenius, Sachi Kabasawa and Akane Odake for their help and assistance, and also to MEXT and JST for their financial support; comments received from reviewers and editors were also indespensible to the writing process.

References

1. Abe, N., Mamitsuka, H.: Query learning strategies using boosting and bagging. In: Proceedings of the Fifteenth International Conference on Machine Learning, vol. 388. Morgan Kaufmann Publishers Inc. (1998)
2. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering **17**(6), 734–749 (2005)
3. Ahn, L.V.: Games with a purpose. Computer **39**(6), 92–94 (2006). DOI 10.1109/MC.2006.196
4. Bailey, R.A.: Design of Comparative Experiments. Cambridge University Press (2008)
5. Balcan, M.F., Beygelzimer, A., Langford, J.: Agnostic active learning. In: ICML '06: Proceedings of the 23rd international conference on Machine learning, pp. 65–72. ACM, New York, NY, USA (2006). DOI http://doi.acm.org/10.1145/1143844.1143853
6. Boutilier, C., Zemel, R., Marlin, B.: Active collaborative filtering. In: Proceedings of the Nineteenth Annual Conference on Uncertainty in Artificial Intelligence, pp. 98–106 (2003). URL citeseer.ist.psu.edu/boutilier03active.html
7. Box, G., Hunter, S.J., Hunter, W.G.: Statistics for Experimenters: Design, Innovation, and Discovery. Wiley-Interscience (2005)
8. Breiman, L., Breiman, L.: Bagging predictors. In: Machine Learning, pp. 123–140 (1996)
9. Bridge, D., Ricci, F.: Supporting product selection with query editing recommendations. In: RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems, pp. 65–72. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1297231.1297243
10. Carenini, G., Smith, J., Poole, D.: Towards more conversational and collaborative recommender systems. In: IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces, pp. 12–18. ACM, New York, NY, USA (2003). DOI http://doi.acm.org/10.1145/604045.604052
11. Chan, N.: A-optimality for regression designs. Tech. rep., Stanford University, Department of Statistics (1981)
12. Cohn, D.A.: Neural network exploration using optimal experiment design **6**, 679–686 (1994). URL citeseer.ist.psu.edu/article/cohn94neural.html
13. Cohn, D.A., Ghahramani, Z., Jordan, M.I.: Active learning with statistical models. Journal of Artificial Intelligence Research **4**, 129–145 (1996)
14. Dagan, I., Engelson, S.: Committee-based sampling for training probabilistic classifiers. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 150–157. Citeseer (1995)
15. Danziger, S., Zeng, J., Wang, Y., Brachmann, R., Lathrop, R.: Choosing where to look next in a mutation sequence space: Active learning of informative p53 cancer rescue mutants. Bioinformatics **23(13)**, 104–114 (2007)

16. Dasgupta, S., Lee, W., Long, P.: A theoretical analysis of query selection for collaborative filtering. Machine Learning **51**, 283–298 (2003). URL citeseer.ist.psu.edu/dasgupta02theoretical.html
17. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences **55**(1), 119–139 (1997)
18. Fujii, A., Tokunaga, T., Inui, K., Tanaka, H.: Selective sampling for example-based word sense disambiguation. Computational Linguistics **24**, 24–4 (1998)
19. Greiner, R., Grove, A., Roth, D.: Learning cost-sensitive active classifiers. Artificial Intelligence **139**, 137–174 (2002)
20. Harpale, A.S., Yang, Y.: Personalized active learning for collaborative filtering. In: SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 91–98. ACM, New York, NY, USA (2008). DOI http://doi.acm.org/10.1145/1390334.1390352
21. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1), 5–53 (2004). DOI http://doi.acm.org/10.1145/963770.963772
22. Hinkelmann, K., Kempthorne, O.: Design and Analysis of Experiments, Advanced Experimental Design. Wiley Series in Probability and Statistics (2005)
23. Hofmann, T.: Collaborative filtering via gaussian probabilistic latent semantic analysis. In: SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pp. 259–266. ACM, New York, NY, USA (2003). DOI http://doi.acm.org/10.1145/860435.860483
24. Huang, Z.: Selectively acquiring ratings for product recommendation. In: ICEC '07: Proceedings of the ninth international conference on Electronic commerce, pp. 379–388. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1282100.1282171
25. Jin, R., Si, L.: A bayesian approach toward active learning for collaborative filtering. In: AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence, pp. 278–285. AUAI Press, Arlington, Virginia, United States (2004)
26. John, R.C.S., Draper, N.R.: D-optimality for regression designs: A review. Technometrics **17**(1), 15–23 (1975)
27. Kapoor, A., Horvitz, E., Basu, S.: Selective supervision: Guiding supervised learning with decision-theoretic active learning. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), pp. 877–882 (2007)
28. Kohrs, A., Merialdo, B.: Improving collaborative filtering for new users by smart object selection. In: Proceedings of International Conference on Media Features (ICMF) (2001)
29. Leino, J., Räihä, K.J.: Case amazon: ratings and reviews as part of recommendations. In: RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems, pp. 137–140. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1297231.1297255
30. Lomasky, R., Brodley, C., Aernecke, M., Walt, D., Friedl, M.: Active class selection. In: In Proceedings of the European Conference on Machine Learning (ECML). Springer (2007)
31. McCallum, A., Nigam, K.: Employing em and pool-based active learning for text classification. In: ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning, pp. 350–358. San Francisco, CA, USA (1998)
32. Mcginty, L., Smyth, B.: On the Role of Diversity in Conversational Recommender Systems. Case-Based Reasoning Research and Development pp. 276–290 (2003)
33. McNee, S.M., Riedl, J., Konstan, J.A.: Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: CHI '06: CHI '06 extended abstracts on Human factors in computing systems, pp. 1097–1101. ACM Press, New York, NY, USA (2006). DOI http://doi.acm.org/10.1145/1125451.1125659
34. Nakamura, A., Abe, N.: Collaborative filtering using weighted majority prediction algorithms. In: ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning, pp. 395–403. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998)
35. Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A., Riedl, J.: Getting to know you: learning new user preferences in recommender systems. In: IUI '02: Pro-

ceedings of the 7th international conference on Intelligent user interfaces, pp. 127–134. ACM Press, New York, NY, USA (2002). DOI http://doi.acm.org/10.1145/502716.502737
36. Rashid, A.M., Karypis, G., Riedl, J.: Influence in ratings-based recommender systems: An algorithm-independent approach. In: SIAM International Conference on Data Mining, pp. 556–560 (2005)
37. Ricci, F., Nguyen, Q.N.: Acquiring and revising preferences in a critique-based mobile recommender system. IEEE Intelligent Systems **22**(3), 22–29 (2007). DOI http://dx.doi.org/10.1109/MIS.2007.43
38. Rokach, L., Naamani, L., Shmilovici, A.: Pessimistic cost-sensitive active learning of decision trees for profit maximizing targeting campaigns. Data Mining and Knowledge Discovery **17**(2), 283–316 (2008). DOI http://dx.doi.org/10.1007/s10618-008-0105-2
39. Rokach, L. and Maimon, O. and Arbel, R., Selective voting-getting more for less in sensor fusion, International Journal of Pattern Recognition and Artificial Intelligence 20 (3) (2006), pp. 329–350.
40. Roy, N., Mccallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: In Proc. 18th International Conf. on Machine Learning, pp. 441–448. Morgan Kaufmann (2001)
41. Rubens, N., Sugiyama, M.: Influence-based collaborative active learning. In: Proceedings of the 2007 ACM conference on Recommender systems (RecSys 2007). ACM (2007). DOI http://doi.acm.org/10.1145/1297231.1297257
42. Rubens, N., Tomioka, R., Sugiyama, M.: Output divergence criterion for active learning in collaborative settings. IPSJ Transactions on Mathematical Modeling and Its Applications **2**(3), 87–96 (2009)
43. Saar-Tsechansky, M., Provost, F.: Decision-centric active learning of binary-outcome models. Information Systems Research **18**(1), 4–22 (2007). DOI http://dx.doi.org/10.1287/isre.1070.0111
44. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 253–260. ACM, New York, NY, USA (2002). DOI http://doi.acm.org/10.1145/564376.564421
45. Schohn, G., Cohn, D.: Less is more: Active learning with support vector machines. In: Proc. 17th International Conf. on Machine Learning, pp. 839–846. Morgan Kaufmann, San Francisco, CA (2000). URL citeseer.ist.psu.edu/schohn00less.html
46. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)
47. Settles, B., Craven, M.: An analysis of active learning strategies for sequence labeling tasks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1069–1078. ACL Press (2008)
48. Settles, B., Craven, M., Friedland, L.: Active learning with real annotation costs. In: Proceedings of the NIPS Workshop on Cost-Sensitive Learning, pp. 1–10 (2008)
49. Settles, B., Craven, M., Ray, S.: Multiple-instance active learning. In: Advances in Neural Information Processing Systems (NIPS), vol. 20, pp. 1289–1296. MIT Press (2008)
50. Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Computational Learning Theory, pp. 287–294 (1992). URL citeseer.ist.psu.edu/seung92query.html
51. Sugiyama, M.: Active learning in approximately linear regression based on conditional expectation of generalization error. Journal of Machine Learning Research **7**, 141–166 (2006)
52. Sugiyama, M., Rubens, N.: A batch ensemble approach to active learning with model selection. Neural Netw. **21**(9), 1278–1286 (2008). DOI http://dx.doi.org/10.1016/j.neunet.2008.06.004
53. Sugiyama, M., Rubens, N., Mueller, K.R.: Dataset Shift in Machine Learning, chap. A conditional expectation approach to model selection and active learning under covariate shift. MIT Press, Cambridge (2008)
54. Swearingen, K., Sinha, R.: Beyond algorithms: An hci perspective on recommender systems. ACM SIGIR 2001 Workshop on Recommender Systems (2001).

55. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. In: P. Langley (ed.) Proceedings of ICML-00, 17th International Conference on Machine Learning, pp. 999–1006. Morgan Kaufmann Publishers, San Francisco, US, Stanford, US (2000). URL citeseer.ist.psu.edu/article/tong01support.html
56. Yu, K., Bi, J., Tresp, V.: Active learning via transductive experimental design. In: Proceedings of the 23rd Int. Conference on Machine Learning ICML '06, pp. 1081–1088. ACM, New York, NY, USA (2006). DOI http://doi.acm.org/10.1145/1143844.1143980