Interactive Machine Learning in Data Exploitation

Reid Porter, James Theiler, Don Hush

Intelligence and Space Research Division

Los Alamos National Lab, USA

Abstract

The goal of interactive machine learning is to help scientists and engineers exploit more of their specialized data in less time. Interactive machine learning focuses on methods that empower domain experts to control and direct machine learning tools from within the deployed environment, whereas traditional machine learning does this in the development environment. This difference allows interactive machine learning systems to be more responsive, more accurate and cheaper to develop and maintain. This article provides a basic introduction to the main components and tries to untangle the many ideas that must be combined to produce practical interactive learning systems. It also describes recent developments in machine learning that have significantly advanced the theoretical and practical foundations for the next generation of interactive tools.

Introduction

In a variety of science and engineering applications the quantity of data being collected far exceeds our capacity to digest and interpret that data. Machine learning can help in these applications by providing tools that clean up, filter and identify the most relevant subsets of data. The broad applicability of these tools is a strength, but because they lack domain-specific input, they are not as accurate or as robust as they could be, and domain experts do not fully trust them.

There are two main ways to make machine learning tools more accurate and useful in specific applications. The most common approach employs computer scientists, or knowledge engineers who have a sophisticated understanding of both the application and the tools, to translate and encode what they learn from domain experts. This is a time-consuming and expensive process, but the importance of science and engineering datasets often justifies the up-front investment.

Interactive machine learning provides a second way to tailor machine learning to specific applications. The central idea is to engage end-users directly and to provide data visualization and annotation tools that enable experts to customize the tools for their application. This approach is particularly attractive in applications where objectives are harder to define up-front, scientific hypotheses are subject to change, or cost /time constraints mean solutions that are good enough today are preferred over solutions that will be optimal tomorrow.

Interactive machine learning ideas and techniques have been pursued in a variety of forms since machine learning began. These efforts come from several research communities including analysis, visualization, and human-computer interaction and deal with a number of different technical issues

including learning, interfaces, programming and interaction. The first objective of this article is to untangle some of these components to better understand where the state-of-the-art is, and where it is going. Towards this goal, this article is divided into 3 parts corresponding to the axes of a 3 dimensional design space for interactive machine learning systems (see Fig. 1). The second objective of this article is to introduce readers to new developments along these axes, and describe how they enable new kinds of interactive machine learning tools.

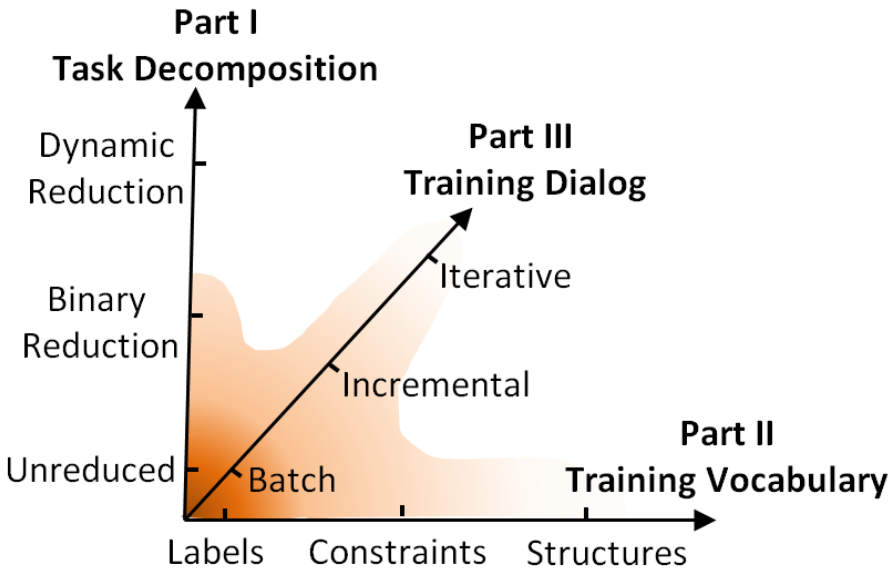


Figure 1: Traditional machine learning has developed important methods that lie at the origin of this design space. Interactive machine learning builds upon advances in the training vocabulary and the training dialog to produce more sophisticated and dynamic task decompositions between humans and machines.

Interactive Machine Learning Design Space: Machine learning tools are used within end-user environments in a number of different ways. In traditional machine learning applications (the origin in Fig. 1), tasks are unreduced and humans and machines work independently on different tasks. The computer is responsible for the automated tasks, and the user is responsible for different, manual tasks. In interactive machine learning, humans and computers begin to work together on the same task. For example, the most common decomposition divides the task into two parts and the human and the computer each do a part. We call this aspect of interactive machine learning 'task decompositions' (Part I).

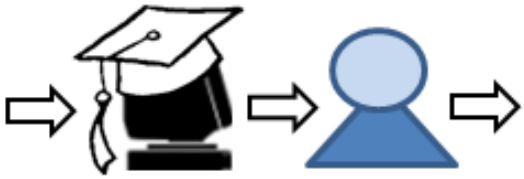
One of the unique features of machine learning is *training*: the ability to optimize tool performance using examples of desired results. In traditional machine learning applications, the algorithm designer uses a fixed set of labels, supplied up-front by the domain experts, to train machine learning tools before they are deployed on the larger dataset or data archive. This approach (Batch, Labels in Fig. 1) and has proven successful both in theory (rigorous proofs) and in practice (commercial applications). But recent developments in machine learning show that there is an opportunity to enhance the generality, efficiency and accuracy of the training process. We have moved from label learning to structure learning, formalized in terms of clusters, constraints, matches and other generalizations of standard

labels. We call these aspects of training the 'training vocabulary' (Part II). We have also moved from up-front learning to on-line, incremental and active learning paradigms that formalize an increasingly interactive dialog between users and their data. We call these aspects of training the 'training dialog' (Part III).

Interactive machine learning is not for everyone. Consumer electronics are driving large numbers of applications where the end-users are consumers and must be enticed into providing feedback and training data for machine learning systems. The priority in these applications is to minimize the interaction [1]. This makes optimizing tools more difficult, but these applications are not critical, and the mistakes made by these tools do not have significant consequences. However, science, engineering, health and defense datasets are specialized and applications are often critical. Domain experts in these applications are also highly motivated and willing to interact with data, and tools, in a much more sophisticated way than current systems allow. These are the applications where interactive machine learning can have the most impact, and these are the applications that will drive interactive machine learning tool development.

Part I – Task Decompositions

Machine Learning Predicts and Humans do the Rest



One of the most common task decompositions involves computers in a first stage identifying subsets of data that are relevant to humans in a second stage. This is a common decomposition because computers can do what they do best – analyze large volumes of data - and the users can do what they do best – analyze a small subset of data in more detail. In this section we describe a number of different tasks where this decomposition is common. Often this decomposition comes about because the task is just too complex for computers to get it right, and so humans are required to validate, clean-up and correct the results.

Content Detectors: A very important task in machine learning is to identify relevant subsets of data. We use the term *relevant* very broadly and it could represent anything from faces in imagery to trends in the stock market. This task is formulated as a machine learning problem by associating a variable $y \in \{+1, -1\}$ with each data sample x that indicates that the sample is relevant (+1) or not (-1). The challenge is to find a function f that predicts y and makes a small number of mistakes:

$$e(f) = \mathbb{E}[f(x) \neq y] \tag{1}$$

The (Bayes) optimal detector f^* is the function (from the space of all possible functions, $\mathcal{F}$) that minimizes Eq. 1:

$$f^* = argmin_{\forall \mathcal{F}}\, e(f) \tag{2}$$

This detector can also be interpreted as a hypothesis test between a model of relevant data $P_{rel}(x)$ verse a model non-relevant data $P_{non}(x)$. In this formulation, the solution to Eq. 2 can be expressed in terms of a likelihood ratio test:

$$f^*(x) = \frac{P_{rel}(x)}{P_{non}(x)} > 1$$

Mistakes come in two flavors, and in interactive applications, these are often presented as *precision*--the probability that data predicted +1 is actually relevant, and the *recall*--the probability that all the relevant data was identified. A detector with precision equal to 1 would have no false alarms (FA). A detector with recall equal to 1 would have no missed detections ($1 - \mathrm{P}_{MD}$). Of course in practice, due to any number of factors (including noise, background variability and approximations made by the machine learning methods) detectors will almost certainly have non-zero number of false alarms and missed detections.

In the interactive setting it is common to cue (or present) the user with the positive predictions so that they can perform triage and/or additional analysis on the reduced data volume. The user would like to process as much relevant data as possible, but their time and effort is limited, and so a useful variation of the standard design criteria (1) is to limit the number of false alarms. For example in fraud detection the classification system often has no utility unless the false alarm rate can be kept below a fixed level. This is called a Neyman-Pearson, or constant false alarm rate, design criteria [2]:

$$f^* = argmin_{\forall \mathcal{F}}\, \mathrm{P}_{MD} \quad \text{subj.} \quad \mathrm{P}_{FA} < \alpha$$

Anomaly Detection: In some applications the relevant content of interest is not known in advance or it is difficult to define formally, but users still need help exploring large datasets. In this case machine learning methods have been developed which use more general models of relevance and non-relevance. Anomalies are defined as subsets of the data with low likelihood. Anomaly detectors can be designed by finding functions with the following form:

$$f^*(x) = \frac{U(x)}{P(x)} > 1$$

where $P(x)$ is the probability density function (i.e. likelihood) for data sample x and $U(x)$ is the uniform density [3]. Various alternatives to $U(x)$ have been suggested and may provide more informative models of relevance such as a product distribution of the data marginals [4, 5]:

$$f^*(x) = \frac{\prod_i P(x_i)}{P(x)} > 1$$

Anomalous Change Detection: This model of relevance has particular utility when the factorization is appropriate to the application. One application where this idea has been developed in detail is change detection, where data typically falls into two (or more) sets. For example, imagine we are looking for (anomalous) change between two spectral signatures taken at two different points in time $x_t = [x_{t,1}, x_{t,2}, \dots, x_{t,D}]$ and $x_{t'} = [x_{t',1}, x_{t',2}, \dots, x_{t',D}]$. A suitable detector would be [6]:

$$f^*(x) = \frac{P(x_t)P(x_{t'})}{P(x_t, x_{t'})} > 1$$

Rare Category Detection: A variation of the anomaly detection problem is rare category detection (RCD). It is motivated by the intuition that in addition to having low-probability, content of interest will form small clusters (or categories). This *clustering assumption* implies we are often more interested in small groups of anomalies than we are in single isolated anomalies. This has also led to the concept of cluster (or category) discovery: users only need to see one example from each category and the objective is to maximize the number of categories shown to the user while minimizing the number of examples shown. This is in contrast to traditional anomaly detection, where the objective is to identify all anomalies as accurately as possible. This is illustrated in Fig. 2.

Theory work has called the RCD problem "multiple output identification" and has characterized the separable case [7]. A number of metrics for quantifying the RCD assumptions in more practical settings have been suggested including the maximum change in local density [8], quantities associated with boundary points [9] as well as a number of quantities derived from hierarchical mean shift clustering [10]. In previous work we suggested the detector:

$$f^*(x) = \frac{P(x)}{P^s(x)} > 1$$

Where $P^s(x)$ is a smoothed (or smoother) density estimate of the data compared to $P(x)$ [11]. In practical settings, once a user has discovered a set of exemplars, they often want to find additional examples of particular categories. This second (exploitation) stage has been called rare category characterization [12].

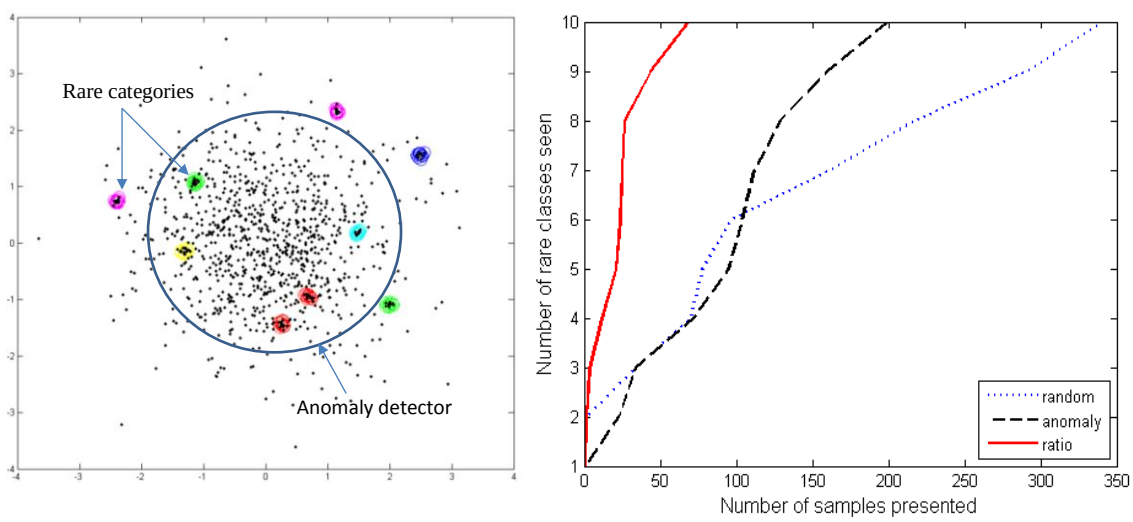


Figure 2: On the left is an example of a synthetic rare category detection problem. Nine rare categories (circled) are embedded within a larger background category. The rare category detection objective differs from anomaly detection because the detector is only rewarded for one example from each category. Detectors tuned to this objective (ratio) outperform anomaly detection which identifies isolated low probability samples (all samples outside of the circle in the left panel).

Humans Sketch and Machine Learning does the Rest

The second task decomposition is the complement of the first and this time, the human goes first. It has proven successful in several applications where the task is typically easy for a human to partially (or approximately) complete, but to fully complete the task would be tedious, expensive and error prone. The decomposition benefits both parties: the human does less work since they only have to provide a partial solution. The computer can obtain higher accuracy compared to automated methods since its computation can be focused on the relevant task. This decomposition is seen in a number of applications which we briefly describe.

Labeling data: A number of data exploitation tasks can be reduced to semi-automatic labeling of data. A human provides labels (e.g. +1/-1) for a subset of the data, and the computer predicts labels for the unlabeled data. In this section we focus on transductive learning [13], which means we are given the unlabeled data at the same time as the labeled data, and the labels are not generalized beyond this fixed dataset. In theory (and practice) there is often not a clear distinction between transductive and the more common inductive methods, which we call *learning by example* in the next section [14]. In interactive settings, however, this distinction is often very clear. In the transductive case, users are required to interact with each new dataset to generate new labels. In the inductive case, the interaction can be generalized to future datasets, so that it is possible (in principle) for the user to stop labeling after the first dataset. A number of popular techniques in semi-supervised learning are transductive [15] and there are a number of applications where transductive methods are preferred.

For example, imagine a quality control application (see Fig. 3). There is too much variability in the production processes from one day to the next to design an automated inspector. But within each day, there is enough consistency between parts for the job to become repetitive and tedious. In terms of interactive machine learning there are several open questions with respect to transductive methods. For example, how should users balance their time between providing labels up-front, validating and correcting the inevitable mistakes that the learning system will make? This depends on a number of application specific factors such as the relative cost of labeling data, verse validating results, verse correcting results. A simple illustration of this tradeoff is illustrated on the right in Fig. 3.

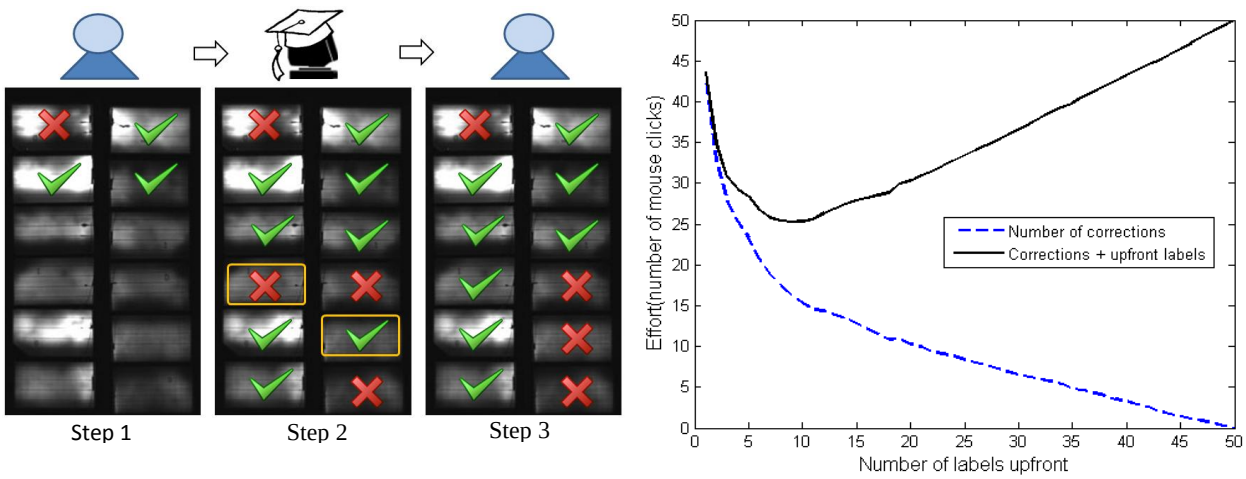


Figure 3: Left) A user needs to inspect 12 solar cells from the production line and identify which cells pass quality control. The user labels 4 cells (left) and then machine learning predicts labels for the remaining 8 (middle). Validating the results, the user identifies two samples that must be corrected (right). Right) Results on a synthetic problem showing the relationship between up-front effort and the total effort required when the cost of validation is small compared to the cost of labeling and correcting.

Clustering data: A large part of understanding data is being able to group subsets of the data into conceptual clusters. Humans are very good at identifying trends and important features of data in one, two to three dimensions, but they need help dealing with the many hundreds of dimensions, and multiple data types, that are often associated with real-world data. Organizing and visualizing clusters is key to a large number of interactive data analysis tools that are motivated from a visual analytics perspective [16]. These tools enable users to efficiently browse and explore large volume datasets with intuitive graphical user interfaces [17].

Interactive machine learning methods can also help in these applications by optimizing the clustering strategy based on a small amount of user input. The typical user interaction is formalized as equivalence constraints: pairs (or sets) of data that belong to the same cluster and/or pairs (or sets) that belong to different clusters [18]. These constraints can be obtained from the user through labeling interfaces, or, through *drag-and-drop* type interfaces where user's visualizing clusters are able to drag subsets of data closer to other subsets [19].

Solution methods for constrained clustering have been developed for both transductive and inductive settings. In the transductive setting the constraints are assumed to be specific to a cluster and there is no attempt to generalize the constraints to other clusters. When a new clustering problem is presented, the user provides new constraints. One of the first transductive methods extended K-Means [20]. We discuss inductive versions of this problem in the next section with respect to training by example.

Image segmentation: Similar to clustering, the goal of image segmentation is to partition data (in this case, pixels) into a number of disjoint contiguous sets. Unlike the general clustering problem, image segmentation typically deals with low-dimensional data with strong local dependencies. But with these

(often significant) differences in mind, interactive image segmentation can be considered one of the most successful applications of constrained clustering.

User input is typically generated by *paint program* like tools, producing paintbrush strokes shown in Fig. 4, which identify pixels that should belong to the same segment (or cluster). Different brush strokes are assumed to belong to different segments, and this generates non-equivalence constraints. Interactive image segmentation is an important tool in bio-medical imaging for the accurate delineation of complex 3-dimensional objects such as organs and bones [21], identifying synaptic pathways [22], as well as identifying, counting and characterizing different cell types [23]. Similar applications are also found in material science [24], geology, manufacturing and food inspection.

A large number of techniques have been developed for interactive image segmentation and many are inspired by transductive semi-supervised clustering techniques [25]. The strong local dependencies also mean that many methods are formulated in terms of Graphical Models, or energy minimization on a graph. Graphical models are briefly introduced in Fig. 4. Couprie et. al. show how graph-cuts [26], random walkers [27], watershed [28], geodesic [29] and many other interactive image segmentation systems can be understood as different parameter choices of a general energy function [30]. The importance of interaction in segmentation problems has also started to motivate frameworks for evaluating performance that don't require expensive human subject testing [31].

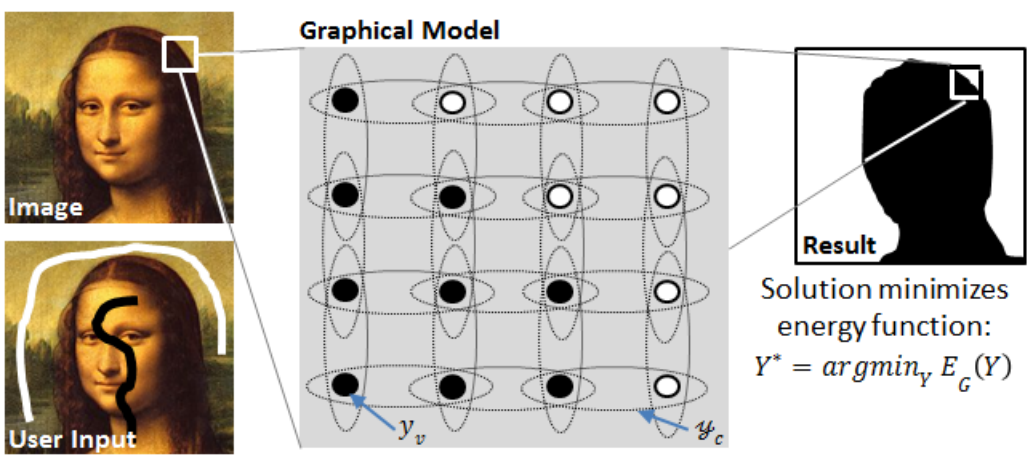
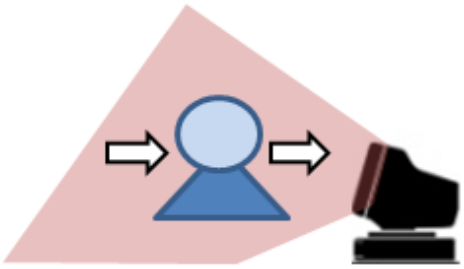


Figure 4: A graphical model is a graph $G = (\mathcal{V}, \mathcal{C})$ defined by a set of vertices $\mathcal{V}$ and cliques $\mathcal{C}$, a set of random variables Y, and a real-valued energy function $E_G(Y)$. Each vertex $v \in \mathcal{V}$ indexes a random variable $y_v \in Y$. In image segmentation the vertices lie on a lattice corresponding to the image plane, and each variable indicates whether that location is part of the background (white) or the foreground (black). Each clique $c \in C$ indexes a subset of the random variables, $\mathcal{Y}_c = \{y_i, y_j, \dots, y_k\}$. In the example, we illustrate pairwise cliques and the energy function has the form, $E_G(Y) = \sum_{c \in \mathcal{C}} \mathrm{f}_c(\mathcal{Y}_c)$, where feature functions, f_c, are chosen for the application. In image segmentation, pairwise feature functions assign low energy to smooth variable assignments based on the magnitude of the image gradient. This means, that the optimal assignment, $Y^* = argmin_Y E_G(Y)$, produces a spatially contiguous labeling that is consistent with strong edges (as shown in the hypothetical result).

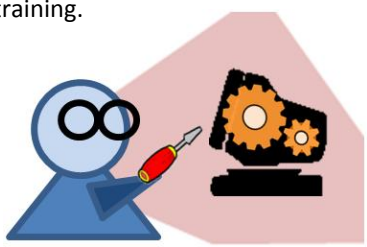
Part II - The Training Vocabulary

So far we have described decompositions where humans and computers work together on tasks with relatively static job assignments. In this section we begin to describe the interactions between humans and machines during training. Training can be used prior to deployment to improve the computer's role in the decompositions described in Part I. It can also be used in the deployed environment to make interactive systems more dynamic: systems can optimize their performance based on data and interactions, and thereby improve over time. The potentially dynamic nature of the training interaction is described in Part III of this article. In this section we focus on the vocabulary used during training and describe the two main categories of interaction shown in Fig. 5.

Figure 5: Two main categories of interaction used in training.



Learning from Examples: This is the most common interaction. The key idea is to collect examples of humans performing a task, and then use the examples to train a computer to perform the task.



Specialized Programming: The second form of interaction amounts to humans programming the computer to perform the task. Several interactive machine learning efforts can be considered specialized programming languages that enable users (who may not necessarily know how to program) to optimize performance.

Learning from Examples

Labeling Data: In Part I we introduced content detectors, and then briefly described their application to labeling data in a transductive semi-supervised setting. In this section we describe the more common setting, which is inductive supervised learning. Training assumes a user provides labels for a randomly selected subset of the data. The training set includes N (data, label) pairs:

$$S = \{ \big(x(1), y(1)\big), \big(x(2), y(2)\big), \dots, (x(N), y(N))\}$$

from which we can define a training set (or empirical) error:

$$\hat{e}(f) = \frac{1}{N}\sum_{n=1}^{N} f\big(x(n)\big) \neq y(n) \tag{3}$$

Learning from examples involves finding a function (from a space of functions, $\mathcal{F}$) that minimizes a version of empirical error that provides good performance guarantees.

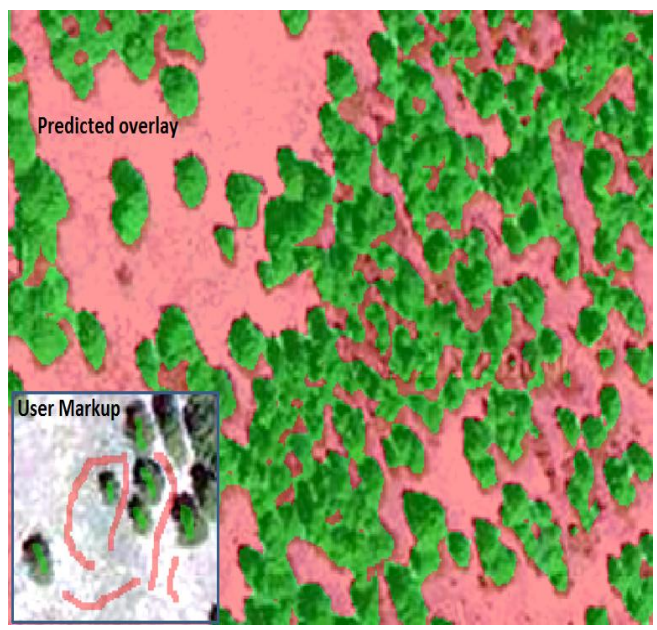
$$\hat{f} = argmin_{f \in \mathcal{F}}\, \hat{e}(f) \tag{4}$$

Performance guarantees relate the performance of $\hat{f}$ in Eq. 4 to the (Bayes optimal) performance f^* in Eq. 2 and help us understand things such as 1) *consistency:* given infinite samples does the performance of $\hat{f}$ converge to the performance of f^*, and 2) *sample complexity:* how many training samples are required for $\hat{f}$ to reach performance within ε of f^*. When we learn functions using this method, we can also use them to predict labels on future data, assuming that it is drawn from the same distribution as the training set.

As a concrete example of how this approach is used in an interactive setting, we turn to image processing, and the task of labeling pixels. This application is the basis of the Crayons interactive machine learning system [32] as well as our own Genie image exploitation system [33]. Both of these systems obtain training examples from users through paintbrush-like tools, and then use supervised learning methods to develop a pixel classifier that can be applied to the larger image or image archive. This is illustrated in Fig. 6.

Figure 6: The paintbrush user-interface is similar to Fig. 4, but represents a completely different form of interaction. Previously, the brush strokes were used as *seeds* from which algorithms would *grow* segments. The number of brush-strokes was equal to the number of final segments.

In the figure to the right, the brush strokes in the lower left are translated into pixel labels. The number of brush strokes is not important, just the total number of labeled pixels. The labels are used to train a pixel labeler, which can then be applied to a much larger image or image archive.



Semi-supervised Learning: Inductive learning methods are also applied in semi-supervised settings, and the unlabeled data (in addition to the labeled training set) often helps detector performance. In some interactive settings it is only possible (or convenient) to collect examples from one of the classes. Often this is the target class, and this variant of semi-supervised learning has been called learning from positive and unlabeled examples [34, 35]. In other applications, such as anomaly detection, the labeled class can correspond to data which is known to be uninteresting, or nominal [36, 37].

Metric Learning: Inductive machine learning methods have also been developed for constrained clustering. In this case, methods are often called metric learning, which highlights the fact that constraints are used to optimize a distance measure which can then be used with standard algorithms (such as K-means) to improve performance on future datasets. Fig. 7 (adapted from [38]) illustrates the concept. Many extensions of this basic idea have been developed, both parametric [39] and non-parametric [40]. A Bayesian treatment of the problem incorporates the clustering constraints within a prior [41]. A method called Bayesian Visual Analytics (BaVA) integrates a projection component for visualization purposes and implements a drag-and-drop interface for generating soft constraints [42].

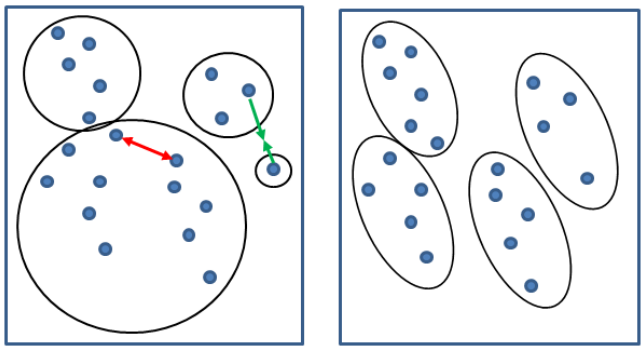


Figure 7: Left) A hypothetical clustering from a Euclidean distance based method (such as K-means) produces an initial clustering of the data. A user identifies equivalence (green) and non-equivalence (red) constraints. Right) A parameterized Mahalanobis distance metric optimized to satisfy the constraints. The modified distance metric generalizes constraints to other clusters.

Structured Output Prediction: Over the last ten years, training methods for image segmentation have advanced rapidly in a sub-field of machine learning called *structured output prediction*. This field (and others) have extended *learning by example* from labels (and constraints) to sequences, trees, segmentations [43-45] and other complex objects [46]. Training structured output predictors proceeds much like label learning, except the training examples are much more complicated. For example, when labeling pixels each training example involved a single pixel and its corresponding label. When learning to segment, each training example involves a complete image and its corresponding segmentation. This is illustrated in Fig. 8

In general, structured training data is complex and collecting it from users in an interactive setting is non-trivial. In addition, the structure is typically fixed in advance and only approximates reality, which means training data is often not intuitive. This has limited the applicability of structured prediction methods to applications where the training data investment can be justified. For example, the "I2T: Image-To-Text" research project at UCLA leverages an association with the LOTUS Corporation, where 10 full time employees generate the necessary training data [47]. Researchers are beginning to address this problem and methods that enable structured training data to be collected from users in more interactive applications are beginning to appear [48-50]. In [51] the authors developed a HMM based system for learning sequences of mouse clicks used by humans to delineate shapes in imagery. More recently, an interactive approach to image-level semantic labeling of images was proposed [52].

Learning to Imitate: As structured prediction grows more ambitious, new connections are being made to hard problems in robotics [53], where the problem is called imitation learning [54] or behavioral cloning [55]. Imagine you are trying to program a computer to drive a vehicle. The control problem is

difficult for humans to get right with programming alone. Imitation learning provides a complementary approach, where humans provide examples by driving the vehicle through remote cameras and controls. In this case, the examples that they generate are *trajectories* through a state-space. These trajectories are used by inverse reinforcement learning methods to optimize cost / reward functions such that the computer's trajectories are closer to the examples [56-58]. Imitation learning has produced a number of general purpose techniques that are starting to be adapted to more interactive training paradigms for data exploitation tasks [59].

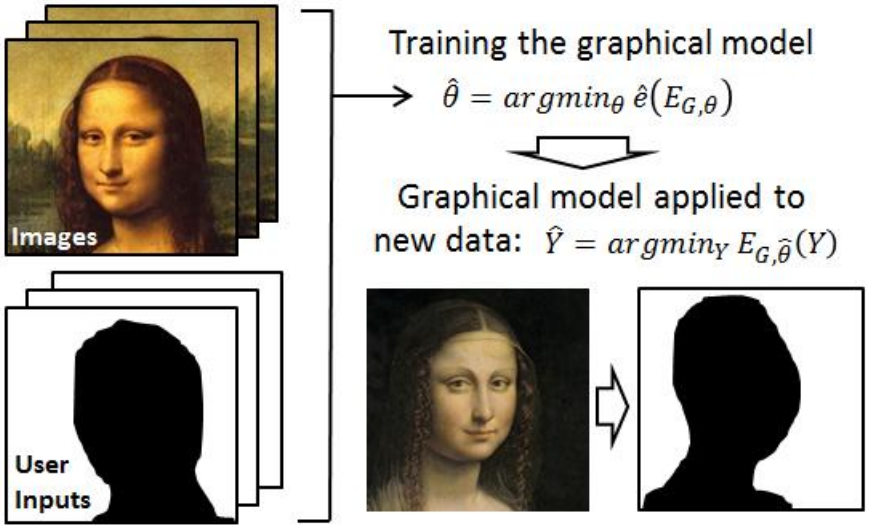


Figure 8: Structured output prediction is a generalization of label learning ($y \in \{+1, -1\}$) to a vector output space, $Y \in L^D$, $L \in \{1,2, \dots, k\}$, that encodes complex structures. In label learning, the content detectors are simple functions: $y = f(x)$. In structured output prediction, content detectors have the form: $Y = argmin_Y E_G(Y)$, where $E_G(Y)$ is the energy function defined in Fig. 4. Much like label learning, we introduce parameters into the graphical model energy function, $E_{G,\theta}(Y)$, and then uses IID training examples to find optimal values for these parameters.

Specialized Programming

The second main form of human-computer interaction during training is *programming*. Several early works in interactive machine learning can be considered specialized programming languages that enable users, who are not programmers, to optimize solutions. More recent work has developed a number of more general purpose methods for summarizing and visualizing machine learning results so that users can intuitively steer learning systems to meet application-level objectives.

Graphical languages: A common framework for early efforts in interactive machine learning involves a scatter plot visualization of data and direct (or indirect) manipulation of decision boundaries. Decision trees are a popular method for this type of training [60] because they are fairly easy for most users to understand (a decision is made by applying a number of rules), and they treat high-dimensional datasets as a series of 1-dimensional features, which means they are relatively easy to visualize. Tools for designing Bayesian networks have also benefited from these characteristics although the probabilistic

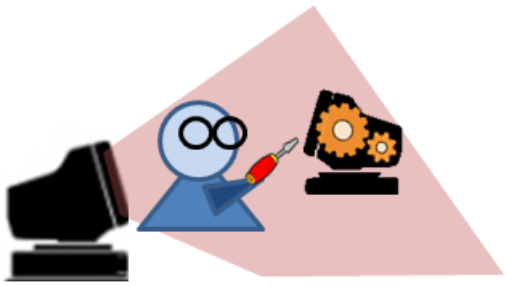
rules used in the tree are more difficult to understand [61]. The approach has also been developed for more complex models such as Hidden Markov Models for bio-informatics applications [62]. The performance of user-specified models has been shown to be competitive to *learning by example* in some applications [63]. However, although these tools are often extremely useful to machine learning researchers, they generally have not moved outside of the research environment.

Some things are best left to humans: Other forms of specialized programming are very widespread and have become standard components in many *learning by example* applications. They typically focus on a small set of parameters that: 1) are difficult to estimate from examples, and 2) are understandable to users through intuitive visualization tools. For example, thresholds are difficult to set automatically because they often depend on prior probabilities that can change from one dataset to the next. Thresholds are also easy for users to understand and manipulate with sliders, particularly if the impact of the threshold on the result can be visualized in real-time. This approach is used extensively in labeling and segmentation applications [64].

When users adjust thresholds in binary classification problems, they are also indicating they prefer one type of error over another e.g. users care more about false alarms than missed detections. This type of interaction can be generalized to the multi-class setting by enabling users to interact with the classifier's confusion matrix. The EnsembleMatrix method uses the confusion matrix to provide real-time feedback on performance while users adjust the parameters and structure of an ensemble classifier [65]. The ManiMatrix approach enables users to directly specify class specific preferences by adjusting weights on the confusion matrix itself [66].

Feature selection is another critical component in real world applications, and it is often up to the user to decide which features should be used in the machine learning system. Interactive data visualization tools can help users make these decisions up-front [67]. Researchers have also developed methods to incorporate user feedback on feature relevance into the training method [68]. This is motivated by the observation that labeling features (as relevant or not relevant with respect to a target class) is often easier for users than labeling examples. A general purpose learning framework, called generalized expectation, has been used to exploit feature labeling in classification [69] and structured output prediction settings [70]. Note, that this framework could be categorized equally well as *learning by example* with what the authors call *weakly labeled data*.

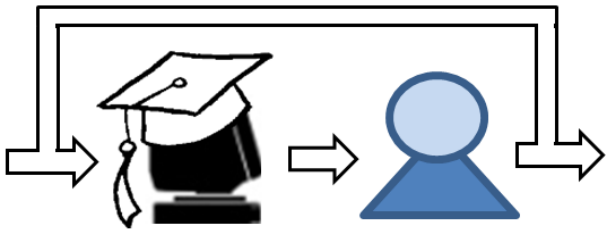
Learning from humans programming computers: In many of the programming techniques we have described, the user interaction is not learnt or applied to future datasets. The final *program* can be applied to new data, but the programming itself is specialized. But there are situations where the programming can become examples for a meta-learning system. For example, a user may apply a



pixel classifier to a number of images and adjust the final threshold each time, and there may be relationships between the image content and the threshold that could be exploited. This type of approach has been used to automate user interactions observed with the EnsembleMatrix method [71].

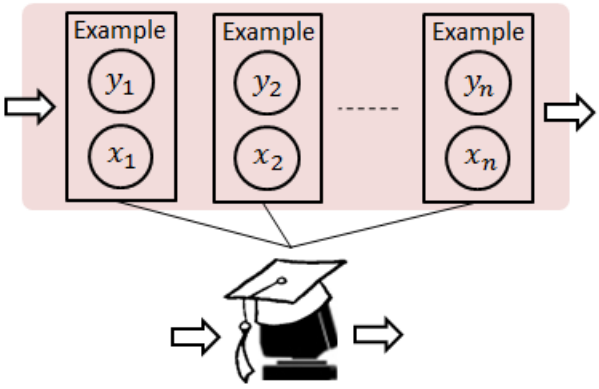
For the programmers: Domain specific languages for machine learning continue to be developed as machine learning components mature and machine learning applications grow. In almost every language, developers are realizing that machine learning provides key functionality that must be supported This includes open source languages, such as Haskell [72] , as well as commercial languages, such as .NET [73]. Although these languages are too specialized to be used by end-users today, they will be a key technology for building (and scaling up) the interactive learning systems of the future.

Part III - The Training Dialog



So far, we have discussed common task decompositions and outlined the growing vocabulary that humans and computers use during training. In this section we describe the different interactive dialogs that can emerge between humans and computers during training.

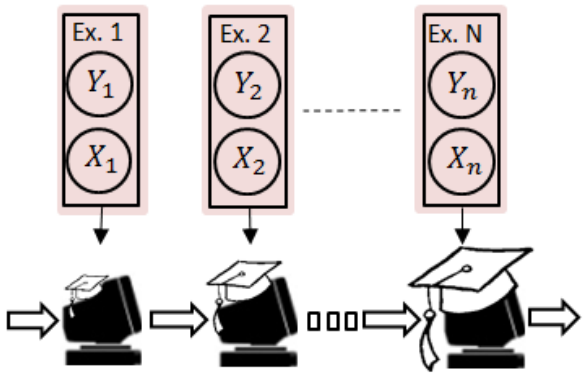
The Monologue



The standard (or default) formulation of the training interaction is *batch learning* where all training examples are provided up front. The examples could be labels, pairwise constraints, sequences, or any of the complex structures described in Part II. In all cases, examples are assumed to be drawn independently from the same underlying distribution (IID: Independent and Identically Distributed). Training methods take these examples and typically solve convex optimization problems to find models with low probability of error. A large body of theoretical work provides performance bounds for this problem setting [13, 74].

The dialog between end-users and computers during batch learning is very simple: 1) The user generates examples using a data browser with annotation tools, and 2) the computer ingests these examples into the learning algorithm.

One Word at a Time



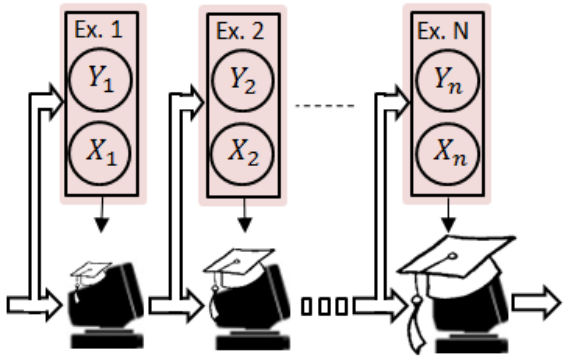
A different formulation of training provides examples to the computer incrementally: one sample at a time, or more generally, a subset at a time. This is called *online learning* and the approach has significant computational advantages when there are a large number of examples. Theoretical developments show that online learning methods can obtain performance bounds within $O(\ln N)$ of batch methods [75-77]. Practical developments have produced some of the fastest and most scalable learning methods for support vector machines [78] and structured output prediction [79].

Most work on online learning assumes that although examples are introduced sequentially, they are still IID. Recent work has started to relax this assumption [80] and this will help guide method development for interactive learning systems where users monitor and provide periodic examples of data with temporal dependencies such as time-series data feeds.

Long-term learning: Although not directly related to online learning methods, incremental learning strategies are also important to a long-term interactive dialog. Computers are becoming increasingly pervasive, and user interaction has rapidly changed from single person interactions over short periods of time with no history, to persistent multi-person interactions over long periods of time.

It will take some time for unified theories of training at multiple scales to emerge, but researchers are beginning to prototype frameworks [81], and practical applications are already being deployed. A general approach in these long-term learning systems is to manually partition the accumulated examples into sets appropriate to the application. For example in content based search, examples have been integrated through separate short-term and long-term components [82]. In a commercial email application examples are integrated through separate individual and group preference components [83].

A Conversation



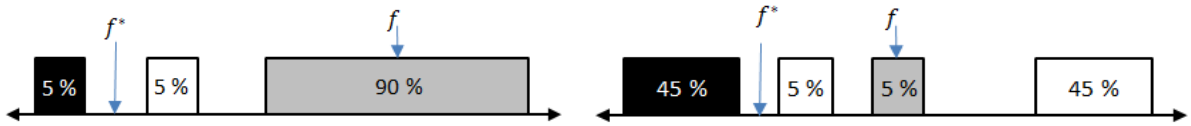
When examples are provided incrementally to a learning system it opens the door to more interactive learning paradigms where there are dependencies between the model learnt at time t, and the training examples provided by the user at time $t + 1$. These dependencies take a number of different forms depending on the interactive setting, and in this section we discuss some of them.

Relevance Feedback: When computers are used as content filters or detectors (as described in Part I) users can often provide feedback (e.g. labels) for predicted results. This concept is used extensively in content based search applications, such as image retrieval, where it is called relevance feedback [84]. In this setting, the typical dialog is:

1. Start with a small number of examples and build a content detector.
2. Apply content detector to unlabeled data and present most relevant samples.
3. User provides labels for samples indicating they are relevant (or not).
4. Update content detector based on the new labels.
5. Goto 2

Relevance feedback has traditionally been motivated by search in semi-structured data, and includes techniques such as query expansion, where text queries are expanded to include things like synonyms and spelling mistakes [85]. Machine learning methods have also been developed, but are often presented in the context of specific applications such as document retrieval [86]. More recently, general purpose machine learning methods and analysis techniques are being incorporated into relevance feedback methods [87]. A fundamental problem in relevance feedback, and interactive dialogs in general, is sampling bias. The samples that the user labels in step 3 are not selected randomly, but the methods used in step 4 assume they are. This means there are no guarantees that query (or detector) performance will get better as more labels are obtained, and in fact, it may get worse. Fig. 9 provides a simple illustration of the sampling bias issues.

Sampling Bias: Two examples, adapted from [88], where data is uniformly distributed in clusters on a line. White blocks contain data with label +1 (i.e. relevant). Black blocks contain data with label -1. Gray blocks contain a 50% mixture of +1 and -1 labels. Content detectors include the space of threshold functions (data to the right of the threshold is predicted +1). These examples show how common sampling strategies for relevance feedback (left) and active learning (right) converge to classifiers with higher error than the detector found through IID sampling (f^*).



Initial samples will (typically) come from the large mixture and the initial detector will be f. In the relevance feedback setting, users are shown the most relevant samples, which are the samples furthest right. Labels for these samples are a mixture of +1,-1 and so the detector does not move.

Again, the initial sample would (typically) place the initial detector at f. In the active learning setting, the user is shown data that is closest to the decision boundary. Labels for these samples are a mixture of +1,-1 and so the detector does not move.

Figure 9: Understanding and mitigating sampling bias is an important first step in developing interactive training dialogs such as relevance feedback and active learning.

Active Learning: Active learning uses a different strategy for selecting examples. It focuses on minimizing the number of labels required to obtain a given level of performance (the sample complexity). Note, that with respect to the end-users application, these strategies may well select the most uninteresting samples in the data set. In some sense, active learning is a training strategy for non-interactive settings: once a user has done the minimal amount of work to build an accurate detector, they can then make productive use of the detector in the application.

A long-standing challenge for machine learning theory has been to develop active learning strategies that perform as well as batch learning, but with fewer samples. Mitigating sampling bias has been a key topic, and a number of methods have been developed that provide safety guarantees and batch learning performance in the worst case [88]. New analysis frameworks have also been required, and new parameters have been developed for active learning performance bounds that enable sample complexity to be quantified and understood [89, 90].

Rare Category Detection: Interactive exploration of large datasets has motivated a large number of iterative learning techniques based on anomaly and rare category detection. The dialog follows a similar format to relevance feedback, but often starts with no examples, and unsupervised detectors in step 1. A number of different strategies for selecting which samples to label have been proposed. Often these strategies include a combination of the most anomalous samples (a relevance feedback strategy), as well as the most ambiguous samples (an active learning strategy) [91, 92]. This mixed strategy arises from the fact that there is often a tradeoff between exploration and exploitation in discovery type tasks. We would like to bring important data to the users attention as quickly as possible but we are unsure

exactly what data is important. Rare category detection has mainly focused on the category discovery problem, and this is a form of exploration. However once categories are discovered, it can be useful to improve the accuracy of detectors for these known categories. This enables users to find more examples of known categories (exploitation tasks such as rare category characterization [12]). But also, if we have a better idea of what we know, it may help us identify what we don't know. This means that even though active learning strategies do not show users what they want to see in the short term, they can lead to detectors that discover more categories with less samples in the long term. A decision theory framework that balances the active learning and rare category objectives was recently proposed [93].

User Bias: So far we have focused on situations where the computer determines which examples to label next based on the previous result. An alternative is for the user to choose which samples to label next. This means that users must be able to visualize (or browse) a larger subset of data and predicted results to make their selections. Empowering users to select the samples can be advantageous since users often know the most important aspects of the problem. In addition, users can sometimes learn the strengths and weaknesses of a learning system through interaction, and then choose examples which guide the system towards better solutions. An example of this phenomenon is shown in Fig. 10. A number of other interactive search applications have been developed that support and benefit from this type of interaction [94]. Theory for how to formally incorporate this type of bias into learning systems is yet to be developed.

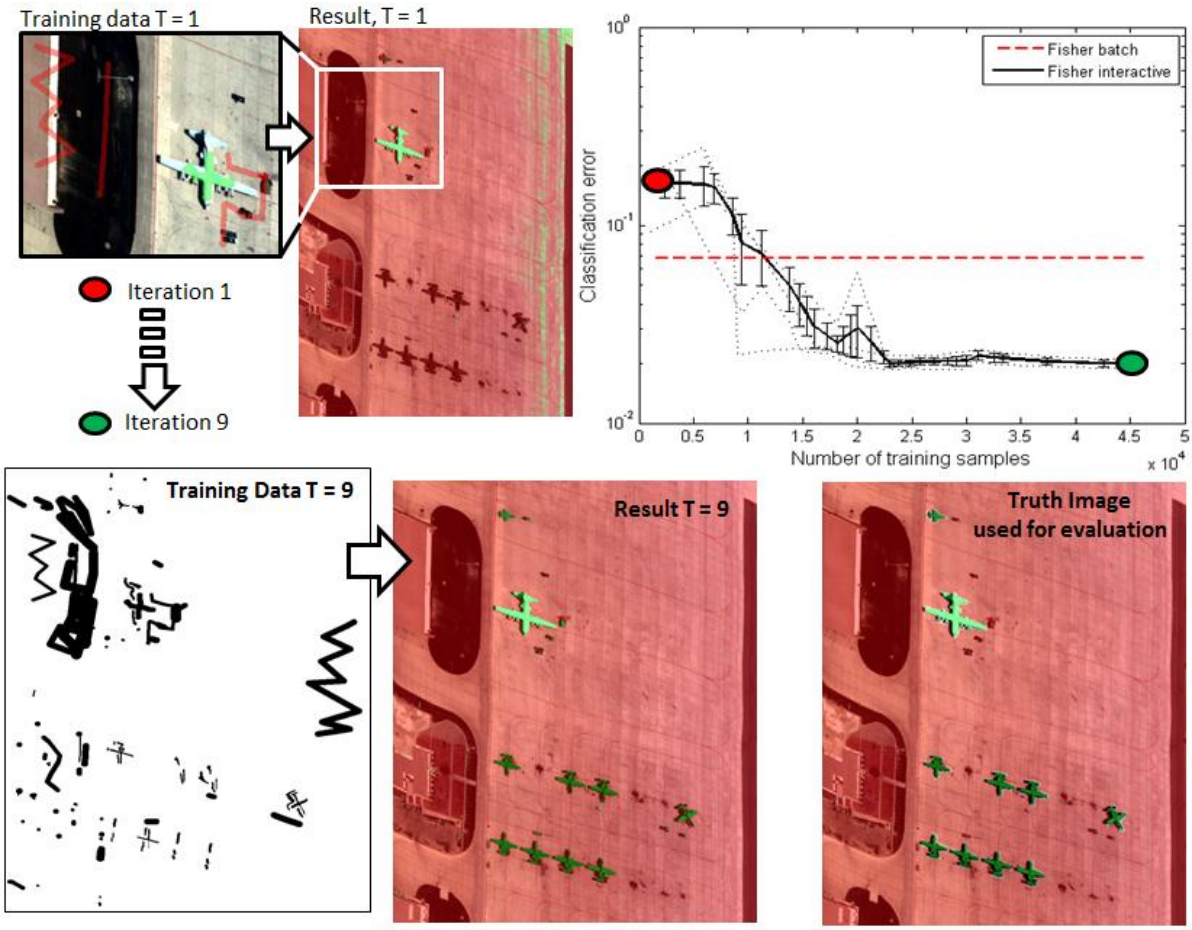


Figure 10: The task (shown in the lower right) is to delineate the airplanes visible on a runway in a ten band multi-spectral image. The red dashed line at 0.07 (upper right) is the performance of Fisher's linear discriminant with IID samples – the classifier has low variance and high bias. In the top left, a user provides a small (biased) subset of labels. The discriminant result (T=1) has an error of 0.2 (red marker). The user inspects the predicted result, provides additional examples, and the discriminant is re-optimized. This process iterates. At the end of the 9th iteration, the user has labeled the data shown in the bottom left and the discriminant result (T=9) has an error close to 0.02. The user performs the experiment 4 times, starting at different parts of the image each time. The average of the 4 experiments is shown in black with error bars and consistently outperforms the discriminant trained with IID samples (green marker).

Future Directions

Interactive machine learning has an exciting future with many open research questions and many opportunities in science and engineering. One area where we expect to see particular progress is in the generality (and pervasiveness) of interactive learning systems. Currently, most interactive learning systems are specialized tools that expect a specific form of interaction and are used in specific parts of the application. But in most applications, users are engaged in a much larger conversation that involves multiple tools and activities such as data preparation and post-processing. In Part II we described the growing vocabulary that has emerged in machine learning and we suggest this will fuel the development

of more general interactive systems. In Fig. 11 we show a typical image analysis workflow used in material science that already uses the entire vocabulary described in Part II.

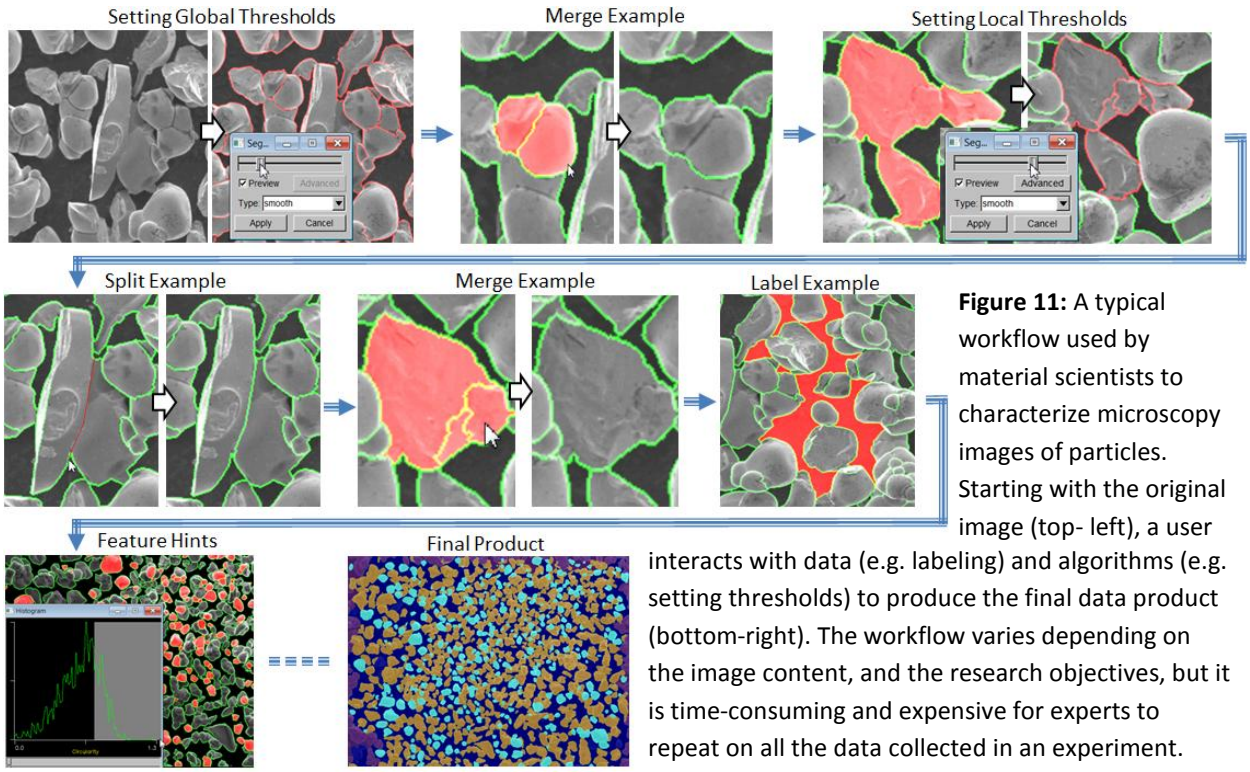


Figure 11: A typical workflow used by material scientists to characterize microscopy images of particles. Starting with the original image (top- left), a user interacts with data (e.g. labeling) and algorithms (e.g. setting thresholds) to produce the final data product (bottom-right). The workflow varies depending on the image content, and the research objectives, but it is time-consuming and expensive for experts to repeat on all the data collected in an experiment.

One of the greatest challenges for interactive machine learning will be the consistent and efficient exploitation of the complex conversation that occurs in workflows like Fig. 11. It is the interaction, more than the vocabulary, that often contains the invaluable domain knowledge that humans possess and machine learning needs [95]. Users are very good at finding ways to use a small set of (often inadequate) tools to reach their objectives, and machine learning can use this to advantage. For example, the final product in Fig. 11 could be used as training data in a structured output prediction system. However this is an extremely complex problem and it is unlikely that existing methods would solve this problem with any reasonable level of accuracy without extensive hand tuning (even if there were large numbers of these examples available). However, it may well be that the sequence of user interactions can provide interactive learning systems with a road-map to efficient solutions.

As part of dealing with unstructured dialogs, interactive machine learning must identify and partition recurring patterns of interaction as examples. Current interactive systems require that the sequence of interactions be manually partitioned into relevant subsets. For example, we could post-process the workflow in Fig. 11 to extract all the merge examples, and then develop tools that predict additional merge candidates [96, 97]. This approach provides a starting point but it ignores temporal dependencies that may provide important clues into the structure of the problem e.g. users may often split and merge segments in sequence, and / or, the user may be using different merging strategies at different points in time.

Another part of dealing with an unstructured dialog is knowing when to automate, and when to engage users. In Part I we described binary task decompositions in which both humans and computers played roles. In Part III we described dialogs in which some of these task decompositions evolve over time as the computer learns to automate, or as humans program new components. We also saw that learning can be applied at multiple levels, and sometimes user interactions are automated and sometimes they are not. A great promise of interactive machine learning is automated task decomposition where the dialog evolves over time and leads to optimized task decompositions for the problem and the resources at hand. It is interesting to note that in other areas of computer science, technologies such as crowd-sourcing are enabling humans to be cost effective in tasks that are traditionally performed by machines [98]. Different groups of humans (and machines) have different skills, and different costs, and this will be an important factor in how tasks are decomposed.

Finally, as humans and machines become more tightly integrated in machine learning, human factors related to user bias and attention come into play [99]. This topic also appears in the context of exploiting crowd-sourced data products and methods to mitigate noise and other issues are an active topic of research [100]. Human factors may well become even more important as interactive machine learning expands the vocabulary and flexibility that users have. As we saw in Part III, user bias can help steer simple classifiers to better solutions, and in fact, bias is critical to interactive machine learning reaching its full potential. Theory for formalizing the positive impacts of bias (domain expertise), as well as methods to mitigate the negative impact of bias (human factors), are in their infancy. In practice, we have yet to see if humans and computers can learn to trust each other in critical applications.

Bibliography

1. Cour, T., et al. *Learning from ambiguously labeled images*. in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. 2009.
2. Cannon, A., et al., *Learning with the Neyman-Pearson and min-max criteria*, 2002, Los Alamos National Laboratory
3. Steinwart, I., D. Hush, and C. Scovel, *A classification framework for anomaly detection.* Journal of Machine Learning Research, 2005. **6**: p. 211-232.
4. Theiler, J. and D.M. Cai. *Resampling Approach for Anomaly Detection in Multispectral Images*. in *Proc. SPIE*. 2003.
5. Abe, N., B. Zadrozny, and J. Langford, *Outlier detection by active learning*, in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*2006, ACM: Philadelphia, PA, USA. p. 504-509.
6. Theiler, J., *Quantitative comparison of quadratic covariance-based anomalous change detectors.* Applied Optics 2008. **47**: p. F12-F26.
7. Fine, S. and Y. Mansour, *Active sampling for multiple output identification.* Machine Learning, 2007. **69**(2-3): p. 213-228.
8. He, J. and J. Carbonell, *Nearest-Neighbor-Based Active Learning for Rare Category Detection*, in *NIPS: Neural Information Processing Systems*2007: Vancouver, B.C., Canada
9. Huang, H., et al., *RADAR: Rare Category Detection via Computation of Boundary Degree*, in *Advances in Knowledge Discovery and Data Mining*, J. Huang, L. Cao, and J. Srivastava, Editors. 2011, Springer Berlin Heidelberg. p. 258-269.

10. Vatturi, P. and W.-K. Wong, *Category detection using hierarchical mean shift*, in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*2009, ACM: Paris, France. p. 847-856.
11. Porter, R., et al., *Toward Interactive Search in Remote Sensing Imagery*, in *SPIE Defense Security and Sensing*2010: Orlando, FL.
12. He, J., H. Tong, and J. Carbonell, *An effective framework for characterizing rare categories.* Frontiers of Computer Science, 2012. **6**(2): p. 154-165.
13. Vapnik, V.N., *Statistical learning theory*, ed. Wiley1998, New York.
14. Chapelle, O., B. Schölkopf, and A. Zien, *A Discussion of Semi-Supervised Learning and Transduction*, in *Semi-Supervised Learning*, O. Chapelle, B. Schölkopf, and A. Zien, Editors. 2006, MIT Press.
15. Zhou, D., et al. *Learning with local and global consistency*. in *Advances in Neural Information Processing Systems 16*. 2004.
16. Thomas, J.J. and K.A. Cook, *Illuminating the Path: The Research and Development Agenda for Visual Analytics*, 2005, IEEE.
17. Cutting, D.R., et al., *Scatter/Gather: a cluster-based approach to browsing large document collections*, in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*1992, ACM: Copenhagen, Denmark. p. 318-329.
18. Basu, S., I. Davidson, and K.L. Wagstaff, eds. *Constrained Clustering: Advances in Algrotihms, Theory and Applications*. Data Mining and Knowledge Discovery Series, ed. V. Kumar2009, Chapman & Hall / CRC.
19. desJardins, M., J. MacGlashan, and J. Ferraioli, *Interactive Visual Clustering for Relational Data*, in *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, S. Basu, I. Davidson, and K.L. Wagstaff, Editors. 2009, Chapman & Hall / CRC. p. 329-356.
20. Wagstaff, K., et al., *Constrained K-means Clustering with Background Knowledge*, in *Proceedings of the Eighteenth International Conference on Machine Learning*2001, Morgan Kaufmann Publishers Inc. p. 577-584.
21. Hahn, H.K. and H.-O. Peitgen. *IWT - Interactive Watershed Transform: A hierarchical method for efficient interactive and automated segmentation of multidimensional grayscale images*. in *Proc. of SPIE* 2003.
22. Jain, V., H.S. Seung, and S.C. Turaga, *Machines that learn to segment images: a crucial technology for connectomics.* Current Opinion in Neurobiology, 2010. **20**: p. 1-14.
23. Wienert, S., et al., *Detection and Segmentation of Cell Nuclei in Virtual Microscopy Images: A Minimum-Model Approach.* Sci. Rep., 2012. **2**.
24. Wang, S., J. Waggoner, and J. Simmons, *Graph-cut methods for grain boundary segmentation.* JOM, 2011. **63**(7): p. 49-51.
25. Duchenne, O., et al. *Segmentation by transduction*. in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 2008.
26. Boykov, Y. and M.-P. Jolly. *Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D images.* . in *International Conference on Computer Vision*. 2001.
27. Grady, L., *Random Walks for Image Segmentation.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2006. **28**(11): p. 1768-1783.
28. Vincent, L. and P. Soille, *Watersheds in digital spaces: an efficient algorithm based on immersion simulations.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1991. **13**(6): p. 583-598.
29. Xue, B. and G. Sapiro. *A Geodesic Framework for Fast Interactive Image and Video Segmentation and Matting*. in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. 2007.

30. Couprie, C., et al., *Power Watershed: A Unifying Graph-Based Optimization Framework.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2011. **33**(7): p. 1384-1399.
31. McGuinness, K. and N.E. O'Connor, *Toward automated evaluation of interactive segmentation.* Comput. Vis. Image Underst., 2011. **115**(6): p. 868-884.
32. Fails, J.A. and J. Dan R. Olsen. *Interactive Machine Learning*. in *Intelligent User Interfaces, IUI '03*. 2003. ACM.
33. Perkins, S., et al., *GENIE - A Hybrid Genetic Algorithm for Feature Classification in Multi-Spectral Images*, in *Proc. SPIE 4120* 2000. p. 52-62.
34. Elkan, C. and K. Noto, *Learning classifiers from only positive and unlabeled data*, in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*2008, ACM: Las Vegas, Nevada, USA. p. 213-220.
35. Hush, D.R. and R.B. Porter. *Density-based similarity measures for content based search*. in *Asilomar Conference on Signals, Systems and Computers* 2009. Pacific Grove, CA: IEEE.
36. Thompson, D.R., et al., *Semi-Supervised Eigenbasis Novelty Detection.* Statistical Analysis and Data Mining, 2012: p. n/a-n/a.
37. Blanchard, G., G. Lee, and C. Scott, *Semi-Supervised Novelty Detection.* Journal of Machine Learning Research, 2010. **11**: p. 2973-3009.
38. Cohn, D., R. Caruana, and A. McCallum, *Semi-supervised Clustering with User Feedback*, 2003, Cornell University.
39. Xiang, S., F. Nie, and C. Zhang, *Learning a Mahalanobis distance metric for data clustering and classification.* Pattern Recognition, 2008. **41**(12): p. 3600-3612.
40. Hertz, T., A. Bar-Hillel, and D. Weinshall, *Boosting margin based distance functions for clustering*, in *Proceedings of the twenty-first international conference on Machine learning*2004, ACM: Banff, Alberta, Canada. p. 50.
41. Lu, Z. and T.K. Leen, *Pairwise Constraints as Priors in Probabilistic Clustering*, in *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, S. Basu, I. Davidson, and K.L. Wagstaff, Editors. 2009, Chapman & Hall / CRC.
42. House, L., S. Leman, and C. Han, *Bayesian Visual Analytics: BaVA*, in *FODAVA Technical Report*2010.
43. Lafferty, J.D., A. McCallum, and F.C.N. Pereira, *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*, in *Proceedings of the Eighteenth International Conference on Machine Learning*2001, Morgan Kaufmann Publishers Inc. p. 282-289.
44. Taskar, B., *Learning Structured Prediction Models: A Large Margin Approach*, 2004, Stanford University.
45. Tsochantaridis, I., et al., *Large Margin Methods for Structured and Interdependent Output Variables.* J. Mach. Learn. Res., 2005. **6**: p. 1453-1484.
46. Vembu, S., *Learning to predict combinatorial structures*, 2010, University of Bonn.
47. Yao, B.Z., et al., *I2T: Image Parsing to Text Description.* Proceedings of the IEEE, 2010. **98**(8): p. 1485-1508.
48. Schwing, A.G., et al. *Efficient Structured Prediction with Latent Variables for General Graphical Models*. in *Int.'l Conf. on Machine Learning (ICML)*. 2012.
49. Huynh, T.N. and R.J. Mooney. *Online Structure Learning for Markov Logic Networks*. in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD 2011)*. 2011.
50. Getoor, L. and C.P. Diehl, *Link mining: a survey.* SIGKDD Explor. Newsl., 2005. **7**(2): p. 3-12.
51. Caelli, T., A. McCabe, and G. Briscoe, *Shape tracking and production using hidden Markov models*, in *Hidden Markov models: applications in computer vision*2002, World Scientific Publishing Co., Inc. p. 197-221.

52. Zankl, G., Y. Haxhimusa, and A. Ion, *Interactive Labeling of Image Segmentation Hierarchies*, in *Pattern Recognition*, A. Pinz, et al., Editors. 2012, Springer Berlin Heidelberg. p. 11-20.
53. Daum\, H., et al., *Search-based structured prediction.* Mach. Learn., 2009. **75**(3): p. 297-325.
54. Schaal, S., *Is imitation learning the route to humanoid robots?* Trends in cognitive sciences, 1999. **3**(6): p. 233-242.
55. Bain, M. and C. Sammut, *A Framework for Behavioural Cloning*, in *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, July 1995]*1999, Oxford University. p. 103-129.
56. Maes, F., L. Denoyer, and P. Gallinari, *Structured prediction with reinforcement learning.* Mach. Learn., 2009. **77**(2-3): p. 271-301.
57. Ross, S., G.J. Gordon, and J.A. Bagnell, *No-Regret Reductions for Imitation Learning and Structured Prediction.* CoRR, 2010. **abs/1011.0686**.
58. Ratliff, N., D. Silver, and J.A. Bagnell, *Learning to search: Functional gradient techniques for imitation learning.* Autonomous Robots, 2009. **27**(1): p. 25-53.
59. He, H., H.D. III, and J. Eisner. *Imitation Learning by Coaching*. in *Neural Information Processing Systems (NIPS)*. 2012.
60. Ankerst, M., et al., *Visual classification: an interactive approach to decision tree construction*, in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*1999, ACM: San Diego, California, United States. p. 392-396.
61. Sucar, L.E. and M. Martínez-Arroyo, *Interactive structural learning of Bayesian networks.* Expert Systems with Applications, 1998. **15**(3–4): p. 325-332.
62. Dai, J. and J. Cheng, *HMMEditor: a visual editing tool for profile hidden Markov model.* BMC Genomics. , 2008
63. Ware, M., et al., *Interactive machine learning: letting users build classifiers.* Int. J. Hum.-Comput. Stud., 2002. **56**(3): p. 281-292.
64. Mangan, A.P. and R.T. Whitaker, *Partitioning 3D surface meshes using watershed segmentation.* Visualization and Computer Graphics, IEEE Transactions on, 1999. **5**(4): p. 308-321.
65. Talbot, J., et al., *EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers*, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*2009, ACM: Boston, MA, USA. p. 1283-1292.
66. Kapoor, A., et al., *Interactive optimization for steering machine classification*, in *Proceedings of the 28th international conference on Human factors in computing systems*2010, ACM: Atlanta, Georgia, USA. p. 1343-1352.
67. Dy, J.G. and C.E. Brodley. *Visualization and Interactive Feature Selection for Unsupervised Data*. in *In Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*. 2000.
68. Raghavan, H., O. Madani, and R. Jones, *InterActive feature selection*, in *Proceedings of the 19th international joint conference on Artificial intelligence*2005, Morgan Kaufmann Publishers Inc.: Edinburgh, Scotland. p. 841-846.
69. Druck, G., G. Mann, and A. McCallum, *Learning from labeled features using generalized expectation criteria*, in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*2008, ACM: Singapore, Singapore. p. 595-602.
70. Druck, G., B. Settles, and A. McCallum, *Active learning by labeling features*, in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*2009, Association for Computational Linguistics: Singapore. p. 81-90.
71. Kapoor, A., et al. *Learning to Learn: Algorithmic Inspirations from Human Problem Solving*. in *AAAI 2012*. 2012. . Toronto.
72. Erwig, M. and S. Kollmansberger, *Probabilistic Functional Programming in Haskell.* Journal of Functional Programming, 2006. **16**(1): p. 21-34.

73. Research, M. *Infer.NET*. 2012; Available from: http://research.microsoft.com/en-us/um/cambridge/projects/infernet/.
74. Bartlett, P., *The sample complexity of pattern classification with neural networks.* IEEE Trans. Inform. Theory, 1998. **44**: p. 525–536.
75. Cesa-bianchi, N., A. Conconi, and C. Gentile, *On the Generalization Ability of On-Line Learning Algorithms.* IEEE Transactions on Information Theory, 2004. **50**: p. 2050–2057.
76. Hazan, E., A. Agarwal, and S. Kale, *Logarithmic regret algorithms for online convex optimization.* Mach. Learn., 2007. **69**(2-3): p. 169-192.
77. Kakade, S.M. and A. Tewari. *On the Generalization Ability of Online Strongly Convex Programming Algorithms*. in *NIPS'08*. 2008.
78. Shalev-Shwartz, S., Y. Singer, and N. Srebro. *Pegasos: Primal Estimated sub-GrAdient SOlver for SVM*. in *In Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML)*. 2007.
79. Ratliff, N., J.D. Bagnell, and M. Zinkevich. *(Online) Subgradient Methods for Structured Prediction*. in *In Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*. 2007.
80. Agarwal, A. and J. Duchi, *The Generalization Ability of Online Algorithms for Dependent Data.* To appear in IEEE Transactions on Information Theory (2012). 2011.
81. Carlson, A., et al. *Toward an Architecture for Never-Ending Language Learning*. in *In Proceedings of the Conference on Artificial Intelligence (AAAI)*. 2010.
82. Xiaofei, H., et al., *Learning a semantic space from user's relevance feedback for image retrieval.* Circuits and Systems for Video Technology, IEEE Transactions on, 2003. **13**(1): p. 39-48.
83. Aberdeen, D., O. Pacovsky, and A. Slater. *The learning behind gmail priority inbox*. in *NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds*. 2010.
84. Rui, Y., et al., *Relevance feedback: a power tool for interactive content-based image retrieval.* Circuits and Systems for Video Technology, IEEE Transactions on, 1998. **8**(5): p. 644-655.
85. Efthimiadis, E.N., *Query expansion.* Annual Review of Information Systems and Technology, 1996. **31**.
86. Rocchio, J., *Relevance feedback in information retrieval*, in *The smart retrieval system—Experiments in automatic document processing*, G. Salton, Editor 1971, Prentice-Hall. p. 313–323.
87. Chen, Z. and B. Fu, *On the complexity of Rocchio's similarity-based relevance feedback algorithm.* J. Am. Soc. Inf. Sci. Technol., 2007. **58**(10): p. 1392-1400.
88. Dasgupta, S. and D.J. Hsu. *Hierarchical Sampling for Active Learning*. in *Twenty-Fifth International Conference on Machine Learning (ICML)*. 2008.
89. Hanneke, S., *A bound on the label complexity of agnostic active learning*, in *Proceedings of the 24th international conference on Machine learning*2007, ACM: Corvalis, Oregon. p. 353-360.
90. Beygelzimer, A., S. Dasgupta, and J. Langford, *Importance weighted active learning*, in *Proceedings of the 26th Annual International Conference on Machine Learning*2009, ACM: Montreal, Quebec, Canada. p. 49-56.
91. Pelleg, D. and A. Moore, *Active Learning for Anomaly and Rare-Category Detection*, in *Proc. 18th Annual Conference on Neural Information Processing Systems*2004.
92. Stokes, J.W., et al., *ALADIN: Active Learning of Anomalies to Detect Intrusions*, 2008, Microsoft Research, MSR-TR-2008-24.
93. Hospedales, T., S. Gong, and T. Xiang, *A Unifying Theory of Active Discovery and Learning*, in *Computer Vision – ECCV 2012*, A. Fitzgibbon, et al., Editors. 2012, Springer Berlin Heidelberg. p. 453-466.

94. Zavesky, E. and S.-F. Chang, *CuZero: embracing the frontier of interactive visual search for informed users*, in *Proceeding of the 1st ACM international conference on Multimedia information retrieval*2008, ACM: Vancouver, British Columbia, Canada. p. 237-244.
95. Pike, W.A., et al., *The science of interaction.* Inf Visualization, 2009. **8**(4): p. 263-274.
96. Porter, R. and C. Ruggiero. *Interactive Image Quantification Tools for Nuclear Material Foreinsics*. in *Proceedings of SPIE*. 2011. San Francisco.
97. Jain, V., et al. *Learning to Agglomerate Superpixel Hierarchies*. in *Proceedings of Neural Information Processing Systems*. 2011.
98. Davis, J., et al. *The HPU (Human Processing Unit)*. in *IEEE CVPR Workshop on Advancing Computer Vision with Humans in the Loop*. 2010.
99. Kobsa, A., *10 Year Anniversary Issue.* User Modeling and User-Adapted Interaction 2001. **11**(1).
100. Tamuz, O., et al. *Adaptively Learning the Crowd Kernel*. in *International Conference on Machine Learning*. 2011.