Performance and Preferences: Interactive Refinement of Machine Learning Procedures

Ashish Kapoor, Bongshin Lee, Desney Tan and Eric Horvitz
Microsoft Research
{akapoor, bongshin, desney, horvitz}@microsoft.com

Abstract

Problem-solving procedures have been typically aimed at achieving well-defined goals or satisfying straightforward preferences. However, learners and solvers may often generate rich multiattribute results with procedures guided by sets of controls that define different dimensions of quality. We explore methods that enable people to explore and express preferences about the operation of classification models in supervised multiclass learning. We leverage a leave-one-out confusion matrix that provides users with views and real-time controls of a model space. The approach allows people to consider in an interactive manner the global implications of local changes in decision boundaries. We focus on kernel classifiers and show the effectiveness of the methodology on a variety of tasks.

Introduction

To date, preferences about the operation of learning and reasoning procedures have been expressed in relatively simple forms, such as "achieve the highest classification accuracy" for a learning task. However, computational procedures for learning and reasoning may generate rich, multiattribute partial and final results (Horvitz 1988), providing opportunities for control and design of learning and reasoning in accordance with human preferences. Rather than seeking to predefine preferences, human assessments of the multiattribute utility of results and behaviors of an automated reasoning system may best be defined in terms of an *interactive exploration of tradeoffs* in the operation of a solver or classifier.

We explore the use of interactive procedures to give system designers or end users the ability to control multiple dimensions of details of the performance of multiclass learning. As an example, in a multiclass setting misclassification costs are often asymmetric and depend on user and task. For example, people may have different preferences about the operation of a junk email filter where a classifier with a low false-positive rate may be preferred over other models given equivalent overall performance. Details about this preference may vary by person.

In general, the performance of classifiers can be described with multiple attributes. The preferences about the performance of classifiers can be encoded as a multiattribute utility function, and can dictate the selection of best models and parameters. When a predefined utility function over different attributes is available, it can guide automated optimization of models and parameters. However, system designers or users may not have access to such assessed preferences. In many cases, detailed preferences might be first defined in the context of specific classification tasks and data: details of the performance of classifiers may interact with goals in unforeseen ways. In such cases, traditional numerical optimization methods cannot help to identify optimal parameters a priori. Preferred behavior might be selected most efficiently via an interactive "dialog" where people have a conversation with a learning procedure about alternate solutions.

We explore methods for allowing people to directly manipulate the operation of machine-learning procedures via a human-in-the-loop methodology. We seek to provide people with tools for exploring tradeoffs that arise with algorithmic procedures, and to give them the ability to refine the operation of learning using an end-to-end analytical pipeline that learns, infers, and provides visualizations about changes in settings at interactive rates. Such assessment of utility or refinements of a prior, coarsely specified utility model in the context of an active learning cycle provides a means of focusing scarce cognitive resources on assessment and control, as relevant choices and tradeoffs are dynamically framed by the operation of the learning algorithm at each cycle.

We focus, as an example, on learning the kernel and hyperparameters for multiclass classification that leverages human guidance. The method enables people to prune the model space via interactive exploration, reducing computational needs. Starting with an initial model, users can interact with a visual representation of a leave-one-out confusion matrix, allowing them to search among a space of models to identify a model whose cross-validation performance is favorably aligned with the desired output. Core technical challenges with developing such interactive methods include composing efficient numerical algorithms that infer settings of hyperparameters at interactive rates.

Background

The performance of a classifier depends on several design choices including the feature set, parametric family, and parameter settings. In the context of kernel-based classification, such choices translate into selecting the appropriate kernel, hyperparameter setting, and choice of regularization parameters. Richer models have been proposed for kernel-based learning with additional hyperparameters, e.g., classification with asymmetric loss (Bach, Heckerman and Horvitz 2006) and automatic relevance detection (Mackay 1992, Neal 1996).

When the utility function is known and easily computable, we can employ techniques such as cross validation to determine the appropriate settings. Cross validation can be used to perform model selection for any kind of classifier and is commonly used given its appealing statistical properties (Evgeniou, Pontil and Elisseeff 2004). However, cross validation can also be prohibitively expensive as it requires classification studies on hold-out sets for all possible models. Further, maximizing simple cross-validation accuracy may not correctly reflect the desired classification output. Consider the case where cross validation shows that multiple models achieve the same best possible performance. In such situations, a model might be selected based on heuristics, such as by the order in which the models are evaluated, where the first model that achieved the best result is selected. Such heuristics may often fail to select an optimal model.

The second set of methods attempts to overcome the computational intensiveness of cross validation by maximizing surrogate functions that reflect appropriateness of the kernel to the observed data. Examples include kernel target alignment for learning SVMs (Cristianini et al. 2001), multiple kernel learning (Lanckriet et al. 2004, Varma and Ray 2007), and evidence maximization using a Bayesian perspective on kernel machines (Girolami and Rogers 2005, Gold, Holub and Sollich 2005, Kapoor et al. 2009). Most of these methods attempt to learn a linear combination of existing kernels and provide no guidance about other parameters. Finally, none of these methods would work when the utility function is unknown, incomputable, or has multiple competing objectives.

We seek to circumvent these challenges via interaction and visualization. The key idea is to harness user interactions to explore the space of solutions without cross validating the entire space in an exhaustive manner. By visualizing the possible solutions and guiding the search, users can both gain a sense of the capabilities of the classifier and choose a model aligned with his goal.

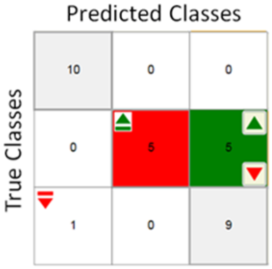
This work comes in the spirit of efforts in *mixed-initiative interaction*, where a computational procedure and user each take initiatives to jointly solve a task (Horvitz 1999, 2007). We extend prior research on interactive machine learning (Fails and Olsen 2003), including studies exploring the value of taking hints from people to optimize the operation of decision trees (Ankerst et al. 1999), naïve-Bayes classifiers (Becker, Kohavi and Sommerfield 2001), SVMs (Caragea, Cook and Honavar 2001), and HMMs (Dai and Cheng 2008). Beyond supervised learning, interactive clustering (Bilenko, Basu and Mooney 2004, Bekkerman et al. 2007) and feature discovery using human input (Raghavan, Madani and Jones 2005) have been proposed. Closest to our work are studies of the use of tools to visualize the performance of specific learning algorithms along with controls for modifying parameters (Ware et al. 2001, Talbot et al. 2008, Kapoor et al. 2010).

Approach

The interactive system consists of two critical components: (1) a user interface (or UI) that provides informative visualizations and enables efficient, intuitive interactions, and (2) a numerical procedure that translates interactions into valid choices of the hyperparameters.

Interactive Confusion Matrix

We employ an interface used in a prior study (Kapoor at al. 2010) that was effective for guiding a simpler interactive machine-learning task. The interactive visualization is a graphical display of a confusion matrix (see Figure 1), that enables users to explore the model space using leave-one-out cross-validation results. The confusion matrix represents leave-one-out classification results, where the row represents an instance's true class and the column predicted one. Depending on the goal, users can increase or decrease the number of instances classified in each cell. The interface allows users to directly interact with this confusion matrix to specify their desire. A click on the up or down arrow that appears on a mouse over a cell corresponds to the desire to increment or decrement the cell value by one. A change in one cell often influences one or more other cells. As it is necessary to control the values in these cells, the system allows users to specify biases (with a ctrl+click) that encourage a specific direction in cell value change. In contrast to the *strong constraint* of incrementing or decrementing a cell, these



10	0	0
0	5	5
1	0	9

Figure 1: Interactive confusion matrix. Users click up/down arrows. Solid arrows indicate desire to increase/decrease a result. Arrows with bar (bias cue) indicate constraints on changes.

biases only define the set of feasible directions until the user clears them.

Given a user's input via the tool, the numerical procedure attempts to find a solution accommodating the request. If a feasible solution is found, the state of the classification is modified and the visualization is updated; otherwise, a notification of difficulty is provided. The changes in the confusion matrix are reflected using colors (green for increase and red for decrease) and opacity (for the magnitude of change). The interface also supports undo (ctrl+z) and redo (ctrl+y). A previous user study on such a confusion matrix showed that users could recover the required parameters faster and more effectively to estimate misclassification risks (Kapoor et al. 2010). We decided to apply the same UI for kernel and hyperparameter learning.

Numerical Procedure to Explore Model Space

Assume that our training set consists of data points $\mathbf{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, with class labels $\boldsymbol{t} = \{t_1, \dots, t_N\}$, and $\boldsymbol{\theta}$ represents the set of hyperparameters that describe our model choice (e.g., kernel width for an RBF kernel, regularization parameters, weights in a linear combination of kernels, etc.). The goal is to determine an updated model $\boldsymbol{\theta}^{new}$ based on user interaction. Given the current model choice, we can run the training algorithm and produce leave-one-out classification results for our training set. We assume a one-versus-all multiclass classification setting; for an input point $\boldsymbol{x}_i$, the leave-one-out multiclass classification results in an *M*-dimensional vector $\boldsymbol{y}_i = [y_{ic}]_{c=1}^{M}$. Here, y_{ic} denotes the classification score for the class *c* and the leave-one-out classification procedure will assign the label that has the highest score among all of the *M* classes. Note that computing these leave-one-out scores for all the training points is an expensive procedure. We address this computational cost in Section 2.3.

As determining a label is performed by max operation, we introduce the notion of an *M*-dimensional *current state vector* denoted as $\boldsymbol{p}_i(\boldsymbol{\theta}) = [p_{ic}]_{c=1}^{M}$, corresponding to the input point $\boldsymbol{x}_i$ and given the model $\boldsymbol{\theta}$. Formally,

$$p_{ic}(\boldsymbol{\theta}) = \frac{e^{\beta \cdot y_{ic}}}{\sum_{c'=1}^{M} e^{\beta \cdot y_{ic'}}}$$

The state vector is simply a softmax transformation of the scores $\boldsymbol{y}_i$, and for a sufficient high value of the scaling parameter β will approximate an indicator vector that is zero everywhere except at the position corresponding to the maximum component of $\boldsymbol{y}_i$.

The leave-one-out confusion matrix can be computed and displayed given the scores $\boldsymbol{y}_i$ for all the training points. A click on the up or down arrow in the confusion matrix expresses the desire for the number of instances in that particular cell to change. In other words, each interaction represents a local intention, leading to the shift of labels for some data points. Clicking on the up arrow on a cell at row a and column b means that at least one more data point with true class a, not already classified as b, needs to be classified as so. Similarly, a down arrow input means that at least one of the points with true class a, and classified as b, needs to be labeled something other than b.

Every interaction defines a set, denoted as U, of data points for which the classification need to change. Further, for every data point $\boldsymbol{x}_i \in U$, the interaction also defines the desired *target* state $\boldsymbol{s}_i$, which is a K-dimensional vector and encodes the desired moves. The desire to classify a point as class b is encoded using a vector, denoted as $\mathbb{I}(b)$, comprising of all zeros except the b^{th} component, which is set to one. Similarly, the desire to not classify a point as b is encoded using $\mathbb{U}(b)$ which is an M-dimensional vector with all entries set to $1/(M-1)$ except the b^{th} component which is set to zero. Thus, given the user interaction, we derive the sets U and the target $\boldsymbol{s}_i$ as:

- Click/Bias Up:
  $U = \{\boldsymbol{x}_i \in \mathbf{X} |\ t_i = a \text{ and } \arg\max\ y_{ic} \neq b\}$;
  $\boldsymbol{s}_i = \mathbb{I}(b), \ \forall\, \boldsymbol{x}_i \in U$
- Click/Bias Down:
  $U = \{\boldsymbol{x}_i \in \mathbf{X} |\ t_i = a \text{ and } \arg\max\ y_{ic} = b\}$;
  $\boldsymbol{s}_i = \mathbb{U}(b), \ \forall\, \boldsymbol{x}_i \in U$
- For rest of the data points unaffected by interaction $\forall\, \boldsymbol{x}_i \notin U, \quad \boldsymbol{s}_i = \boldsymbol{p}_i(\boldsymbol{\theta})$

The goal is to minimize the difference between the current and the target states. Consequently, our strategy is to derive an updated model $\boldsymbol{\theta}^{new}$ that is aligned with user desires. Note that assigning $\boldsymbol{s}_i = \boldsymbol{p}_i(\boldsymbol{\theta})$ for all unaffected acts as a regularizer by implying that we seek a configuration that satisfies the user's preferences but is closest to the original state. Formally, we consider minimization of an objective function that measures discrepancy between the targets and the current state using the *KL* divergence:

$$g(\boldsymbol{\theta}) = \sum_{i=1}^{N} KL(\boldsymbol{s}_i \| \boldsymbol{p}_i(\boldsymbol{\theta})) = \sum_{i=1}^{N} \sum_{c=1}^{M} s_{ic} \log \frac{s_{ic}}{p_{ic}(\theta)}$$

Other choices, such as squared or absolute difference between targets and current state, could be used to measure the disparity. *KL* divergence is a natural choice as both $\boldsymbol{p}_i$ and $\boldsymbol{s}_i$ are valid probability distributions. The gradient of this function can be succinctly written as:

$$\nabla_{\boldsymbol{\theta}} g = -\beta \cdot \sum_{i=1}^{N} [\mathbf{J}_i(\boldsymbol{\theta})]^{\boldsymbol{T}}\, [I - \boldsymbol{p}_i(\boldsymbol{\theta}) \cdot \mathbf{1}^{\boldsymbol{T}}]\, \boldsymbol{s}_i$$

Here, $\mathbf{J}_i(\boldsymbol{\theta})$ denotes the Jacobian matrix comprising of the partial derivation of the score vector $\boldsymbol{y}_i$ with respect to $\boldsymbol{\theta}$ (row c and column d is $\frac{\partial y_{ic}}{\partial \theta_d}$), and $\mathbf{1}$ is a vector of all ones. Since, one click is interpreted as a user's desire to increment the count in the specific cell by one; it is not desirable that we fully minimize the above objective. Consequently, we employ binary search to determine the minimum step-size η that would change the count of the cell where the user incremented the key by one. In case a valid step size is found, we update:

$$\boldsymbol{\theta}^{new} = \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} g$$

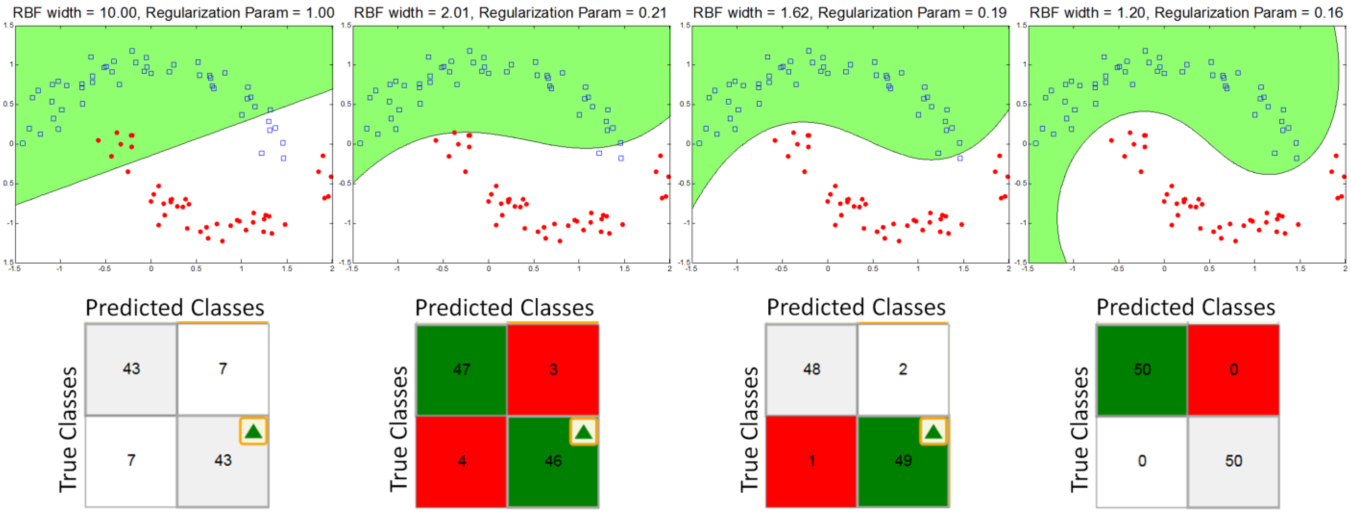


Figure 2: Illustrative example demonstrating learning RBF kernel width and regularization parameter using interactive confusion matrix. Top: Evolution of decision boundary. Bottom: Corresponding state of the confusion matrix.

This procedure can be viewed as a human-in-the-loop search, where the human input determines the descent direction, and the system carries out optimization and computes the correct step size to take. This numerical procedure coupled with interactive visualization and feedback provides a system that users can use to explore the model space. The key to a compelling experience is real-time feedback. Thus, it is imperative to employ numerical operations that allow for updates that are perceived as occurring immediately. The most computationally expensive parts of the numerical procedure are computing the leave-one-out scores $\boldsymbol{y}_i$ and the Jacobian $\mathbf{J}_i(\boldsymbol{\theta})$ that encodes the gradient of scores.

Computational Issues

Many kernel based classification algorithms can be formulated as an optimization problem of finding a solution, $f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})$. Here, $k(\boldsymbol{x_i}, \boldsymbol{x}_j)$ is the kernel that depends upon our model choice $\boldsymbol{\theta}$. In general, the optimization problem takes the follow form:

$$\min \quad \boldsymbol{\alpha}^T K \boldsymbol{\alpha} + \frac{1}{\lambda} \sum_{i=1}^{N} V(t_i, f(\boldsymbol{x}_i))$$

$V(t, f(\boldsymbol{x}))$ represents a loss-function, K is the kernel matrix where the entry, $K_{ij} = k(\boldsymbol{x_i}, \boldsymbol{x}_j)$, and λ is the regularization parameter. Different choices of the loss functions lead to different flavors of methods: e.g., hinge loss ($|1 - f(\boldsymbol{x}_i) \cdot t_i|_+$) for SVMs and squares loss $(f(\boldsymbol{x}_i) - t_i)^2$ for regularized least-square classification (RLSC). Although solving such optimization problems is feasible for large datasets at interactive rates, our requirement goes beyond that. As mentioned earlier, we seek leave-one-out scores, which, if done naïvely require training a classifier N different times. In addition to the leave-one-out scores we need the gradients of these leave-one-out scores with respect to the model $\boldsymbol{\theta}$. For the case of regularized least square classification such leave-one-out scores can be computed efficiently without the need to optimize for N different classifiers. Formally, the leave-one-out outputs and their gradients with respect to θ_d are:

$$f^{loo}(\boldsymbol{x}_i) = t_i - \frac{[A^{-1}\boldsymbol{t}]_i}{[A^{-1}]_{ii}} \text{ , where } A = K + \lambda \cdot I$$

$$\nabla_{\theta_d} f^{loo}(\boldsymbol{x}_i) = \frac{\left[A^{-1}\frac{\partial A}{\partial \theta_d} A^{-1}\boldsymbol{t}\right]_i}{[A^{-1}]_{ii}} - \frac{[A^{-1}\boldsymbol{t}]_i \left[A^{-1}\frac{\partial A}{\partial \theta_d} A^{-1}\right]_{ii}}{[A^{-1}]_{\boldsymbol{ii}}^2}$$

Here, $\boldsymbol{t} = [t_1, .., t_N]^T$ is the vector of all training labels. These computations depend on computing the inverse of A, which is $O(N^3)$. Consequently, the overhead of computing the leave-one-out scores and their gradients is fairly small.

We can also use certain numerical tricks for Gaussian Process (GP) based classifiers (Rasmussen and Williams 2006) in order to compute leave-one-out estimates. Instead of performing an optimization, we perform Bayesian integration and obtain a probability distribution over the set of all possible classifiers. Expectation Propagation (EP) (Minka 2001) is one such approximate inference technique that approximates the distribution of classifiers as a Gaussian whose mean again can be represented as $f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})$. One of the advantages of using EP is that estimates of leave-one-out classification are estimated during the inference process and are available for free. For our interactive procedure we also need the gradients of the leave-one-out estimates. While there is no obvious approximation to these using EP, estimation is numerically feasible due to low dimensionality of the model space.

We ran the system on a 3.00 GHz dual Intel Xeon processor Windows machine with 8 GB RAM. We are able to achieve interactive rates on problems with 3000 data points using RLSC and 400 points for EP. The procedure can be applied to larger problems if the user is willing to tolerate latency in system responses.

Experiments

We now demonstrate the performance of the interactive model exploration system in context of different model selection tasks relevant for multiclass classification. The first two tasks have a well-defined utility function (overall accuracy), thus the experiments show that interactive method is at par with the numerical methods. The third task (asymmetric loss) consists of preferences and performance, highlighting the operation of the method.

Learning Hyperparameters for an Individual Kernel: We first highlight the proposed framework on a non-linear classification task using an RBF kernel. The RBF kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-0.5\ \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/\sigma^2)$ with the kernel width parameter σ is a popular choice for non-linear kernel-based classification. However, the kernel width is an important parameter and the success of the method depends on choosing it correctly. Thus, the model selection task in the context of RLSC is to choose appropriate settings of the hyperparameters $\boldsymbol{\theta} = \{\sigma, \lambda\}$, where λ is the regularization parameter. Although a popular choice is cross validation with grid search, we show how we can avoid that using the interactive method.

Figure 2 illustrates application of interactive model search on a synthetic dataset. Starting with a suboptimal choice of hyperparameters ($\sigma = 10, \lambda = 1$), the user continues to interact with the system until achieving a satisfactory leave-one-out performance. In Figure 2, the top row shows the evolution of the decision boundary as the user interacts, while the bottom row displays the state of the leave-one-out confusion matrix at that instant. Note that the user only sees the confusion matrix and does not have access to views of the decision boundary. The interactive procedure smoothly morphs the initial incorrect model to a correct one which provides a good classification boundary. All this is achieved with just a few clicks from the user and without the overhead of exhaustive search.

We also explore how well interactive model search compares to exhaustive grid search over the model space on real-world data. Table 1 highlights such a comparison with exhaustive grid search on three UCI datasets. We performed exhaustive search on a 2D grid (for kernel width and the regularization parameter) in the range from 0.05 to 5 with a step size for 0.05, where the exhaustive search choose hyperparameters for which leave-one-out accuracy was found to be maximum. Each dataset was split randomly 10 times into a training (60%) and test set (40%). In each trial, the features are normalized to have zero mean and a unit variance using the training set. We report results averaged over 10 different splits (standard error in parenthesis). From the table, we can see that the interactive approach is able to select models that have better or comparable test accuracy to the models found using the grid search: 92.55% vs. 91.63% (Ionosphere), 86.90% vs. 86.31% (Sonar) and 82.48% vs. 82.22% (Heart). Note that grid search only explores model choices at a discrete set of hyperparameter settings (defined by the grid) and there is no such constraint in the interactive procedure. Consequently, we can explain the gain in performance by noting that the interactive procedure can discover models that performed best but were not part of the grid search. Although it is possible to choose a very small step-size for the grid parameter, such finer choices lead to a considerable increase in computational overhead. Table 1 also shows the average of discovered hyperparameters using the two methods. We observe that the solutions discovered by both procedures are fairly close to each other showing that the interactive method can discover better or comparable models without performing exhaustive search. Finally, the interactive method provides significant benefits in terms of time requirements. While

Table 1: Interactively learning σ and λ on UCI datasets.

		RBF width σ	**Regu. Const.** $\boldsymbol{\lambda}$	**Test** **Accuracy**
Ionosphere	*Interactive*	3.48 (0.17)	0.64 (0.06)	**92.55 (0.8)**
	Grid Search	3.95 (0.24)	0.22 (0.04)	91.63 (0.5)
Sonar	*Interactive*	3.54 (0.24)	0.46 (0.31)	86.90 (1.8)
	Grid Search	3.02 (0.31)	0.31 (0.09)	86.31 (1.1)
Heart	*Interactive*	3.12 (0.69)	0.86 (0.25)	**82.48 (0.9)**
	Grid Search	3.20 (0.43)	1.71 (0.15)	82.22 (0.9)

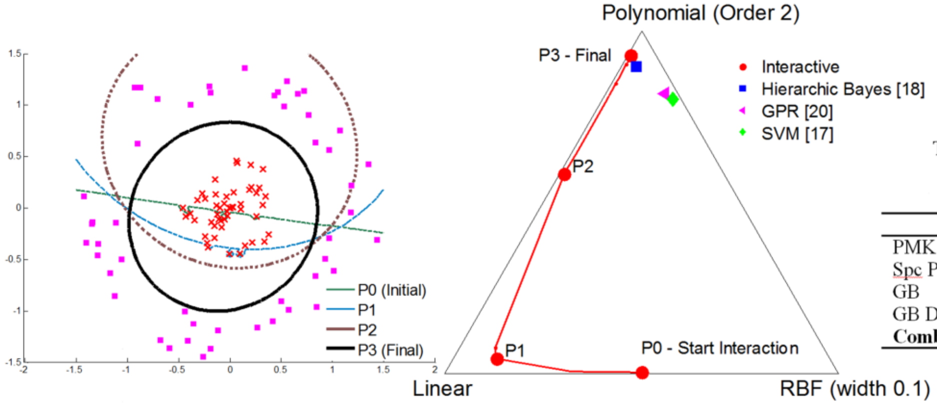


Figure 3: Learning linear combination of kernels. Left: Synthetic classification task, where the evolution of the decision boundary is shown. Middle: Evolution of relative weights of combination in the simplex starting from the bottom-center (equal weights for linear and RBF). P0 to P4 in both left and middle figure correspond to the same four models encountered along the red path. Right: Performance on Caltech-101.

Table 2: Test Accuracy on Caltech-101 using Linear Combination of Kernels

	Interactive	GPR [20]	SVM [17]
PMK	**74.47 (2.8)**	71.93 (3.1)	71.80 (2.1)
Spc PMK	**75.13 (2.4)**	75.13 (2.4)	75.13 (1.6)
GB	**81.53 (1.6)**	81.20 (1.7)	82.20 (1.4)
GB Dist	**79.67 (1.8)**	79.67 (1.8)	82.07 (1.4)
Combine	**86.87 (1.8)**	84.53 (1.6)	82.47 (1.4)

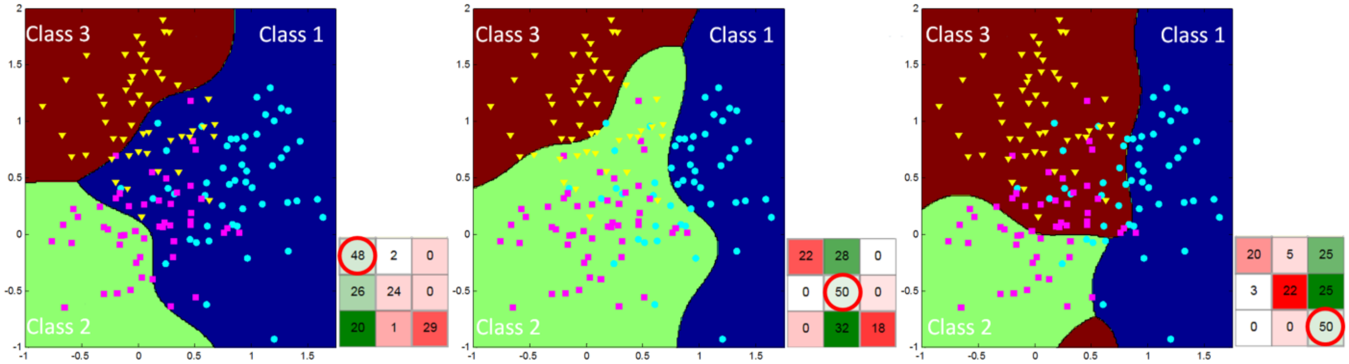


Figure 4: Handling asymmetric misclassification cost. Interactive confusion matrix helps users find parameter settings that result in decision boundaries aligned with misclassification risks. We depict training points, associated confusion matrices converged upon, and resulting decision boundaries. Red circles highlight how users maximized most preferred class.

the exhaustive search took mean times of 111.38, 39.46, and 54.26 seconds (over the 10 splits) for the Ionosphere, Sonar and Heart data, the interactive method only required mean times of 38.12, 20.45, and 32.15 secs, respectively.

Learning Linear Combinations of Kernels: Learning a linear combination of kernels is an extensively studied task (Lanckriet et al. 2004, Varma and Ray 2007, Girolami and Rogers 2005). Formally, given multiple kernels $K_1 .., K_M$, we seek a linear combination of the base kernels such that the resulting kernel $K = \sum_{j=1}^{M} \alpha_j \cdot K_j$ has a good discriminatory power. Besides the regularization constant λ, we also wish to learn the weights $\{\alpha_1 .., \alpha_M\}$.

Figure 3 illustrates the utility of the interactive approach on such tasks. In particular, we generated a synthetic binary classification problem (Figure 3 left) and considered three different basis kernels: a linear kernel, a polynomial kernel, and an RBF kernel with width 0.1. Note, that this synthetic data is perfectly classifiable using a polynomial of degree 2 (a circle). Hence, the ideal kernel combination should assign a high weight on the polynomial kernel. First, we start with weights that had equal weight on the linear and the RBF kernel (1/2 for each) and zero weight on the polynomial kernel. Clearly, such weights are not going to provide good boundary (as data is not linearly separable and the width of 0.1 for RBF kernel is not big enough to provide perfect classification).

Figure 3 (left) also shows how the actual decision boundary evolves with such interactions. The shades of the boundaries represent their recentness, with the darkest boundary representing the final solution; we can observe the smooth transition from earlier solutions as the user interacts with the confusion matrix. Figure 3 (middle) highlights the evolution of weights as the user interacts with the matrix in order to drive the solution aligned with his preference. We illustrate a simplex in Figure 3 (middle), where the vertices correspond to each of the three base level kernels and the interior of the triangle represent relative weights (i.e. $\alpha_j / \sum_{j'=1}^{M} \alpha_{j'}$) of the kernels. The red line illustrates the evolution of the weights with each user interaction starting from the center of the triangle. Note that, naïve cross validation would have required us to explore the whole interior of the simplex. However, with the interactive approach, we can avoid the extra overhead and just follow the user preferences to track a path (red line) leading to the desirable model. We also plot the solutions found by Hierarchic Bayesian learning (Girolami and Rogers 2005), evidence maximization of GP regression (Kapoor et al. 2009), and multi-kernel learning of Varma and Ray 2007. The red line highlights the smooth morphing of the weights to the correct solution, where most of the weight is given to the polynomial kernel. Further, notice that the solution recovered by the interactive procedure is fairly close to the ones found by other techniques, which highlights such interactive methods can are at par with existing methods.

We tested the interactive scheme on learning linear combinations of classifiers for the Caltech-101 task. We considered four different base kernels (see Kapoor et al. 2009): Pyramid Match Kernel (PMK), spatial PMK, Geometric Blur (GB) and GB with the distortion. Since interacting with a 101 class confusion matrix is hard, we only considered classification among 10 classes, which were chosen randomly at each iteration. We consider 15 randomly selected training images per class and use the rest for testing and repeat the process ten times. Table 1 shows the average recognition performance obtained using different methods (standard error in parenthesis). We observe that the kernel combination provides better performance than any classifier based on an individual kernel. Further, the rates obtained by the interactive method (86.87%) are better than other combinations (84.53% for GPR Kapoor et al. 2009 and 82.47% for Varma and Ray 2007). We hypothesize that this results from the fact that, during interactive model selection, we are directly optimizing for leave-one-out error instead of a surrogate function. Also, the mean number of clicks to solve the task over the 10 runs was 10.3 (1.87). This result is even more compelling as the user input only a few clicks to produce classifiers that perform well.

Asymmetric Misclassification Cost and Unbalanced Data: Another flavor of model selection task occurs when misclassification costs vary greatly depending on the outcome (e.g., spam filtering). Such scenarios are addressed by considering separate regularization constants for different classes (Bach, Heckerman and Horvitz 2006). However, given the asymmetric there is no easy way to estimate the settings of regularization parameters. Grid search is again an option, but becomes infeasible as the number of classes increase. Similarly, the unbalanced data scenario can also be handled by considering separate regularization parameters, where estimating such parameterization is non-trivial. However, we can use the interactive procedure to tackle these difficult cases. Figure 4 demonstrates application of interactive model selection on a three-class problem, where the confusion matrix has been guided in three different scenarios that consider each of the three classes as the class with highest misclassification cost. The training points corresponding to the three classes are shown as yellow triangles, blue circles, and magenta squares, and the shading corresponds to the resulting classification boundaries. We observe that by guiding the confusion matrix to the preferred operating point (red circle highlights the entries maximized during interaction), the user can discover models that are aligned with his personal perferences. Rather than searching through changes in the regularization parameters, the user can employ the interactive procedure to translate abstract parameters into real-world consequences.

Summary

We presented methods and results on the interactive optimization of models in the context of kernel-based classification. We showed via a set of experiments how approach can be used to incorporate the preferences of people to prune the search over the large space of possible models. We are excited about the possibilities for developing new forms of interactive optimization of learning and reasoning procedures. We hope the methods we presented will stimulate others to pursue opportunities to develop expressive visualizations coupled with interactive controls that enable people to explore models and parameters for guiding learning and reasoning in accordance with human preferences.

References

Ankerst, M., Elsen, C., Ester, M. and Kriegel, H. P. 1999. Visual classification: an interactive approach to decision tree construction. KDD.

Bach, F., Heckerman, D. and Horvitz, E. 2006. Considering cost asymmetry in learning classifiers. Journal of Machine Learning Research (7).

Becker, B., Kohavi, R. and Sommerfield, D. 2001. Visualizing the simple Bayesian classifier. Information Visualization in Data Mining and Knowledge Discovery. Eds. Fayyad et al.

Bekkerman, R., Raghavan, H., Allan, J. and Eguchi, K. 2007. Interactive clustering of text collections According to a User-Specified Criterion. IJCAI.

Bilenko, M., Basu, S. and Mooney, R. J. 2004. Integrating constraints and metric learning in semi-supervised clustering. ICML.

Caragea, D., Cook, D. and Honavar, V. G. 2001. Gaining Insights into Support Vector Machine Pattern Classifiers using Projection Based Tour Methods. KDD.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A. and Kandola, J. 2001. On kernel-target alignment. NIPS.

Dai, J. and Cheng, J. 2008. HMM Editor: a visual editing tool for profile hidden Markov models. BMC Genomics 9.

Evgeniou, T., Pontil, M. and Elisseeff, A. 2004. Leave one out error, stability, and generalization of voting combinations of classifiers. Machine Learning 55.

Fails, J. A. and Olsen, D. R. J. 2003. Interactive machine learning. IUI.

Girolami, M. and Rogers, S. 2005. Hierarchic Bayesian models for kernel learning. ICML.

Gold, C., Holub, A. and Sollich, P. 2005. Bayesian approach to feature selection and parameter tuning for Support Vector Machine classifiers. Neural Networks, 18(5-6).

Horvitz, E. 1988. Reasoning under varying and uncertain resource constraints. National Conference on Artificial Intelligence.

Horvitz, E. 1999. Principles of Mixed-Initiative User Interfaces. CHI.

Horvitz, E. 2007. Reflections on Challenges and Promises of Mixed-Initiative Interaction, *AAAI Magazine* 28. pp. 19-22.

Kapoor, A., Grauman, K., Urtasun, R. and Darrell, T. 2009. Gaussian processes for object categorization. International Journal of Computer Vision 88(2).

Kapoor, A., Lee, B., Tan, D. and Horvitz, E. 2010. Interactive optimization for steering machine classification. CHI.

Lanckriet,G., Cristianini, N., El Ghaoui, L., Bartlett, P. and Jordan, M. 2004. Learning the kernel matrix with semi-definite programming. Journal of Machine Learning Research (5).

MacKay, D. J. 1992. Bayesian interpolation. Neural Computation (4).

Minka, T. P. 2001. Expectation propagation for approximate Bayesian inference. UAI.

Neal, R. M. 1996. Bayesian learning for neural networks. Lecture Notes in Statistics (118) New York: Springer.

Raghavan, H., Madani, O. and Jones, R. 2005. InterActive feature selection. IJCAI.

Rasmussen, C. E. and Williams, C. K. I. 2006. Gaussian processes for machine learning. The MIT Press.

Talbot, J., Lee, B., Kapoor, A. and Tan, D. 2008. EnsembleMatrix: interactive visualization to support machine learning with multiple classifiers. CHI.

Varma, M. and Ray, D. 2007. Learning the discriminative power-invariance trade-off. ICCV

Ware, M., Frank, E., Holmes, G., Hall, M. and Witten, I. 2001. Interactive machine learning: letting users build classifiers. International Journal of Human-Computer Studies 56(3).