Robust Supervised Learning

J. Andrew Bagnell
Robotics Institute
Carnegie-Mellon University
dbagnell@ri.cmu.edu

Abstract

Supervised machine learning techniques developed in the Probably Approximately Correct, Maximum A Posteriori, and Structural Risk Minimiziation frameworks typically make the assumption that the test data a learner is applied to is drawn from the same distribution as the training data. In various prominent applications of learning techniques, from robotics to medical diagnosis to process control, this assumption is violated. We consider a novel framework where a learner may influence the test distribution in a bounded way. From this framework, we derive an efficient algorithm that acts as a wrapper around a broad class of existing supervised learning algorithms while guarranteeing more robust behavior under changes in the input distribution.

Introduction

It is now widely accepted that there are tasks for which hand-designed algorithms are extremely difficult to engineer but for which humans are able to inject domain knowledge by providing a labeled "training set". Machine learning techniques have broad application in variety of these domains. For instance, recent computer games use a wide assortment of supervised learning techniques; spam-reduction software relies on techniques developed for text classification, and learning results have become instrumental in the development of robotics.

An important commonality of these applications of machine learning algorithms is the violations of the assumptions these algorithms are based on. In particular, classifiers developed in the PAC, Bayesian, and Structural Risk Minimization frameworks all assume that a learner's predictions do not influence the distribution of training examples upon which it will be tested. We discuss the formalization of this restriction below. In practical applications, this limitation of test data distribution matching training data is rarely mentioned explicitly.

Crudely speaking, the goal of supervised learning is to develop algorithms that construct from training data a hypothesis that will not only predict with small error on its training set but demonstrate small test error on future predictions. There are, then, at least two ways in which this can fail. The first, colloquially called "over-fitting" inspires a vast theoretical literature [1] and a great deal of attention in the experimental and applied learning literature.

The second is that in practical application of learning techniques predictions from a learner are often used in a way that directly affects the distribution of examples in the future. In this way, small prediction error, even without overfitting, may lead to a dramatically higher error in actual use.

Practitioners find that ignoring the complication of shifting input distributions can lead to serious difficulties. (Thrun 1995, Pomerleau 1989) Consider the widely-known machine learning success of ALVINN, a neural network designed to drive at speed in both on- and off- road environments. The idea of training a learning algorithms to match a human driver is a simple, elegant approach to the problem. Pomerleau, the principle architect of the learning system, notes however that naive training of the network is insufficient.

Consider for a moment a classifier trained, as ALVINN, to predict steering direction based on input images. We can assume it able to achieve low training set error and so that generalization bounds (e.g. cross-validation) indicate that it will indeed exhibit small error on test-set. As Pomerleau notes, "when driving for itself, the network may occassionally stray from the center of the road so it must be prepared to recover by steering the vehicle back to the center of the road." In reality, a network that makes small prediction error, ϵ, may choose controls that lead to a slightly less typical test example at the next step, which leads to large prediction error and a yet more atypical sample. Pomerleau again, "...problems associated with training on-the-fly stem from the fact that backpropagation requires training data which is representative of the full task to be learned."

In the ALVINN work, Pomerleau developed two

[1] See, for instance, the tutorial of (Langford 2005) covers a variety of methods for ensuring a classifier doesn't overfit.

techniques that explicitly deal with the test-train mismatch. First, Pomerleau develops a system that generated extra training data that effectively emphasizes some of the harder cases ((Pomerleau 1989), section 3.2). [2] Second, Pomerleau ((Pomerleau 1989), section 3.4) uses a buffer to increase training on examples that prove more difficult for the learner.

The techniques described in (Pomerleau 1989) are rather domain-specific, but nevertheless, get to the essence of an important problem. In this work, we provide a specific formalization of the problem of shifting test-set distributions and derive a general algorithm reminiscent of Pomerleau's heuristic that optimally solves the formulation. This formulation allows us to find an optimal solution efficiently and derive meaningful generalization bounds that hold out-of-sample.

Related Work

There are important frameworks that address issues of learners that effect the input distribution. In Reinforcement Learning, we are explicitly interested in agents whose actions affect the world. Recent research in this field can be understood as explicitly trying to control the relationship between test and train distributions. (See for instance, (Bagnell *et al.* 2003) and (Fern *et al.* 2003)). Here we address the fundamentally simpler supervised learning problem where correct outputs are given to us as part of the training set.

The online learning model (Littlestone and Warmuth 1989) makes *no* assumptions about a distribution of inputs. It trades this for making no predictive statements, but instead considering performance in retrospect. Predictive results can be established for many online algorithms but typically make the explicit assumption that the classifier has no effect on the test instances.

One of the closest related works is (Daliv *et al.* 2004) where a game between a learner an agent that can affect the training examples is described. In (Daliv *et al.* 2004), there are more general costs and an adversary can affect different features on instances in the data set. Our framework applies instead to the input *distribution* (and later in the work $p(y|x)$ as well) as opposed to specific examples. It also presents a game that is amenable to efficient optimization algorithms and can make predictive statements.

Robust Learning Framework

In general it is very difficult to predict the changes a classifier will induce on the test-set distribution. It is often reasonable, however, to assume at the outset that the change will be in some way bounded. We would like, then, to do well with respect to any distribution that respects this bound.

Mathematical Preliminaries

We assume that the goal of learning can be phrased, as in Maximum Likelihood, Bayesian Maximum Aposteriori, Structural Risk Minimization, etc, as an expect risk minimization principle. In particular, suppose we have an input space $\mathcal{X}$ and output space $\mathcal{Y}$. Training examples are generated from $\mathcal{D}(x, y) = p_{\text{train}}(x)p(y|x)$. Furthermore, we have a set $\mathcal{H}$ of hypothesis $h : \mathcal{X} \to \mathcal{Y}$ indexed by a continuous parameter vector $\theta \in \Theta$ and a loss function $\ell : \mathcal{Y}, \mathcal{Y} \to [0, 1]$ [3] A common goal in learning theory is to attempt to minimize the *risk* $\mathrm{E_D}[\ell(\mathrm{x}, \mathrm{y}, \mathrm{h}_\theta(\mathrm{x}))]$ over Θ. We'll further make the assumption that Θ is a convex set and that ℓ is concave loss function in θ. These assumption are satisfied by some of the most popular classification and regression techniques (Support Vector Machines, logistic and linear regression, Adaboost, Conditional Random Fields, Naive Bayes, etc.) A weaker notion of local convexity can be used to give similar but weaker results to those developed here for a broader class of learning techniques. In this work, we wish to modify the objective to acknowledge changes in test distribution. Our way of asserting bounded influence of the learner's predictions is to assert that we will be tested on $D'(x, y) = q_{\text{test}}(x)p(y|x)$ where $q_{\text{test}}(x)$ is a distribution over inputs such that $\mathcal{KL}(q_{\text{test}}(x)|p_{\text{train}}(x)) < \epsilon$, where $\mathcal{KL}$ is the relative entropy or KL-divergence:

$$\mathcal{KL}(f|g) = \sum_x f(x) \ln \frac{f(x)}{g(x)}$$

We assume here that the inputs are discrete, although there are simple generalizations to continuous distributions over inputs using continuous relative entropy. (Cover and Thomas 1997) We denote by $\mathcal{Q}$ the set of test distributions that meet the KL constraint.

Relative entropy is a natural information theoretic measure of the difference between distributions that arises throughout machine learning. In particular, information theoretically $\mathcal{KL}(q_{\text{test}}(x)|p_{\text{train}}(x))$ measures the inefficiency in nats incurred for using $p_{\text{train}}(x)$ to develop an optimal code and then testing the code against distribution $q_{\text{test}}(x)$. (Cover and Thomas 1997) Given the requirement of doing well on all such distributions, it is natural to write our requirements as a game

$$\min_{\theta \in \Theta} \max_{q \in \mathcal{Q}} \mathrm{E_q}[\mathrm{g}(\mathrm{x}, \theta)] \quad (1)$$

where

$$g(x, \theta) = \mathrm{E_{p(y|x)}}[\ell(\mathrm{y}, \mathrm{h}_\theta(\mathrm{x}))] \quad (2)$$

[2] This technique actually mixes two benefits: it is a "virtual training example" technique to enforce a particular symmetry, and secondly it emphasizes cases that are rare in the data-set.

[3] The choice of upper loss bound as 1 is arbitrary but convenient. We can drop the finiteness requirement as well although this leads to limitations in proving sample complexity bounds.

in which our learner chooses a classifier so as to minimize its loss while nature chooses $q_{\text{test}}(x)$ so as to maximize the expected loss given knowledge of this classifier.

The game as we have described it has a number of interesting theoretical properties. An important one is the minimax equality.

Theorem 1 *Minimax equality holds in Equation (1) if the set Θ is compact. That is, we have:*

$$\min_{\theta\in\Theta} \max_{q\in\mathcal{Q}} \mathrm{E_q}[\mathrm{g(x,\theta)}] = \max_{\mathrm{q}\in\mathcal{Q}} \min_{\theta\in\Theta} \mathrm{E_q}[\mathrm{g(x,\theta)}]$$

It then does not matter whether nature chooses the test distribution first or the learner chooses its hypothesis first.

Proof: Notice that as ℓ is concave and the objective function is linear, hence convex, in $q_{\text{test}}(x)$, we have a convex-concave saddle point problem. $\mathcal{Q}$ is clearly convex and compact, while by assumption Θ is convex and compact (for example, as a result of Ivanov or Tikhonov style regularization). We appeal then to classic minimax results, e.g. Proposition 2.6.9 of (Bertsekas *et al.* 2003). ■

There are weaker constraints then compactness that let us conclude minimax equality as well, however, compactness is perhaps the simplest as well as the most common occurring in supervised learning.

Solving the game

Next we consider approaches to solving this game. Although we have shown minimax inequality we are considering an optimization problem the dimensionality of which scales with the number of possible inputs. We argue now, using Legendre-Fenchel duality results, that we can solve for the worst case distribution in nearly closed-form.

Theorem 2 *For a fixed hypothesis h_θ, nature chooses distribution $q_{\text{test}}(x)$ to solve the constrained optimization problem:*

$$\max_{q_{\text{test}}(x)\in\mathcal{Q}} E_{q_{\text{test}}(x)}[E_{p(y|x)}[\ell(y,h(x))]] \qquad (3)$$

This problem is optimized by a distribution of form:

$$q_{\text{test}}(x) = \frac{\exp(\beta g(x)) p_{\text{train}}(x)}{Z(\beta)} \qquad (4)$$

with β a scalar and Z the appropriate normalization constant.

Proof: Notice that the objective function is linear in $q_{\text{test}}(x)$ and the constraints on $q_{\text{test}}(x)$ are convex and compact. We can use convex duality then to construct the optimal distribution. (Boyd and Vandenberghe 2004) Form the Lagrangian:

$$\mathcal{L}(q_{\text{test}}(x),\lambda,\gamma) = \mathrm{E_q}[\mathrm{g}] - \lambda(\sum_{\mathrm{x}} \mathrm{q_{test}(x)} \ln \frac{\mathrm{q_{test}(x)}}{\mathrm{p_{train}(x)}} - \epsilon) - \gamma(\sum_x q_{\text{test}}(x) - 1)$$

Solve for the stationary point of the Lagrangian with respect to $q_{\text{test}}(x)$.

$$\frac{\partial\mathcal{L}}{\partial q_{\text{test}}(x)} = g(x) - \lambda(\ln q_{\text{test}}(x) + 1 - \ln p_{\text{train}}(x)) - \gamma = 0$$

gives

$$q_{\text{test}}(x) \propto \exp(\beta g(x)) p_{\text{train}}(x)$$

where β is chosen so that the $\mathcal{KL}$ bound holds. ■

We note that this duality result arises within the robust control community, where the concern is with the control of dynamic stochastic systems, where the dynamics are either ill-specified or vary in some complex way that remains bounded in KL-divergence. Recently, (Nilim and el Ghaoui 2004) exploited this classical result for a particular efficient solution in (Bagnell and Schneider 2001) robust stochastic control framework. It is for this reason that we title our work "robust supervised learning". [4]

Reverse KL-divergence The KL-divergence ordering we have chosen is natural for more reasons than its information theoretic loss interpretation. The reverse KL-divergence, which is also amenable to a similar duality calculation (Nilim and el Ghaoui 2004), intuitively seems unreasonable in our scenario. The reverse divergence, allows test distributions where $q_{\text{test}}(x)$ is *not* absolutely continuous with respect to $p_{\text{train}}(x)$. This makes it a poor model for our goals as the the optimal distribution may place mass on test examples which cannot possible occur in our training data.

Minimizing Empirical Robust Risk

In machine learning problems we cannot, of course directly minimize the true risk function. Instead, we choose some empirical estimate of the risk function, based on the data, to optimize. In our setting, where the input distribution is assumed to change, we must similarly, form an approximation to the risk function. Here we use an importance weighting scheme to affect the change with respect to the empirical distribution. That is, we use the identity

$$\begin{aligned} \mathrm{E_{q_{test}(x)}[g(x)]} &= \mathrm{E_{q_{test}(x)}[\frac{p_{train}(x)}{p_{train}(x)} g(x)]} \\ &= \mathrm{E_{p_{train}(x)}[\frac{q_{test}(x)}{p_{train}(x)} g(x)]} \end{aligned} \qquad (5)$$

to allow us to use samples from $p_{\text{train}}(x)$ to estimate the risk under $q_{\text{test}}(x)$. Our importance weights are then just the ratio of distributions:

$$\frac{\exp(\beta g(x))}{\sum_x \exp(\beta g(x)) p_{\text{train}}(x)}$$

[4] This unfortunately clashes somewhat with the use of "robust" in the statistics community where it refers to models using heavy tailed distributions or particularly flat loss functions.

The boundedness of the KL-divergence and the Radon-Nikodym theorem imply correctness of this change of measure.

An issue now arises regarding computing these weights. If we assume we are in a noise free setting, as in the PAC (Haussler 1995) framework, it is easy to compute a good approximation to these importance weights. We simply evaluate for sample i:

$$w_i = \frac{\exp \beta \ell(y_i, h(x_i))}{\sum_i \exp \beta \ell(y_i, h(x_i))}$$

Unfortunately, if $p(y|x)$ is not almost-surely deterministic, $\exp(\beta l(y_i, h(x_i)))$ may be a badly biased estimate of $\exp(\beta g(x_i)$. We take up further consideration of this point below.

Algorithms

The results above suggest a simple strategy for learning a classifier robust to the classifier own influence on the test distribution. A natural proposition is a dual loop algorithm to solve for the optimal classifier. It is natural to view the optimization problem as $\min_\theta C(\theta)$ where we know C is a concave function that includes the maximization over distributions. This is a rather general convex optimization algorithm and we can apply a host of techniques to including, for instance, subgradient descent type algorithms.

Unfortunately, this approach requires a modification of existing training procedures to accomodate the robust weighting scheme. This may be a good strategy, but generally it does not allow us to take advantage of specialized existing algorithms that have already been developed for supervised learning. To do so, we leverage the minimax inequality (Theorem 1) and propose a dual-loop type algorithm for the dual:

$$\max_{q \in \mathcal{Q}} \min_{\theta \in \Theta} \mathrm{E_q[g(x, \theta)]} = \max_{\mathrm{q} \in \mathcal{Q}} \mathrm{r(q)}$$

One natural approach is projected supergradient ascent. ((Bertsekas *et al.* 2003)) The subgradient supremum theorem tells us that the set of supergradient of r, ∂r includes the set of supergradients of $\partial \mathrm{E_q[g(x, \theta*)]}$ where $\theta*$ minimizes $\mathrm{E_q[g(x, \theta)]}$. We propose a close relative of this approach where instead of choosing a subgradient and projecting, we choose to step in the direction q that maximizes $\mathrm{E_q[g(x, \theta)]}$ for the current θ. This allows us to directly enforce the KL constraints as well as move in a direction closer to the optimal one. This gives us the following algorithm:

Algorithm 1: Dual-loop using supervised learning oracle

Initialize weights $q = [\frac{1}{N} \ldots]$; **repeat**
  $\theta \leftarrow$ Learner(x,y,q);
  $\hat{q} \leftarrow \max_{q \in \mathcal{Q}} \frac{\exp(\beta g(x_i, y_i; \theta))}{\sum_i \exp(\beta g(x_i, y_i; \theta))}$;
  $q \leftarrow \alpha q + (1 - \alpha)\hat{q}$;
until θ *has converged*

It is also natural to try a very simple and fast damped "best-response" type algorithm. Such algorithms in general are difficult to provide convergence guarantees for, but often are very efficient in practice.

Out-of-sample performance

Our approach was derived using the unknown input distribution $p_{\mathrm{train}}(x)$ and then applied using an importance sampling approach on the empirical distribution of examples. It is important to then ask whether performance guarrantees can be expected to hold out-of-sample. By noting that the expectation we care about (under $q_{\mathrm{test}}(x)$) can be written as in Equation (5), we can see that for a fixed β we can prove sample complexity results for testing out- of-distribution. Such bounds, of course, multiply the sample complexity of an underlying algorithm by $O(\exp \beta)$.

Unfortunately, it is also possible to construct examples that show that bounding the performance of a classifier under a change in test distribution that is $\mathcal{KL}$ -bounded by ϵ may require (with high probability) a number of samples that scales exponentially with ϵ. For small ϵ this remains reasonable, and potentially the effective sample complexity on many problems is such that the robustness bound remains applicable.

Experimental Observations

We applied our approach to robust supervised learning to a number of data-sets from the UCI dataset (Blake and Merz 1998) to explore some of its properties. In our experiments we used a maximum-entropy classifier (Nigam *et al.* 1999) for each example (equivalent to logistic regression in the two class instances). We applied Algorithm (1) using the maximum entropy classifier applied directly to the features given in the datasets. To perform a controlled simulation of the effect of applying our classifier to a difficult test distribution, we leveraged the result above on finding the optimal distribution. For an independent test set, we computed the optimal weights, subject to a $\mathcal{KL}$ bound (or equivalently, for a fixed β), and then tested against the resulting weighted data-set.

In Figure (1) we compare a naively trained maximum entropy classifier (i.e. ignoring potential distribution changes) with one trained using the robust algorithm. The figure shows performance on a test set where we have scaled the $\mathcal{KL}$ bound in the same way. (For instance training and testing both a $\mathcal{KL} \leq 0.1$). We measure performance here as the training algorithm does using average log loss. ($\frac{1}{N} \sum_i \log p(y_i|x_i)$)

In the next set of experiments, on the "Iris" data-set, whose results are depicted in Figure (2), we used the same $\mathcal{KL}$ bound (here 0.5) on the test distribution uniformly over all the classifiers. Here we vary instead just the $\mathcal{KL}$ bound applied during training. It is interesting to note, as in Figure (2) right, that the test error often continues to drop when the training $\mathcal{KL}$ bound is substantially greater than the test $\mathcal{KL}$ bound.

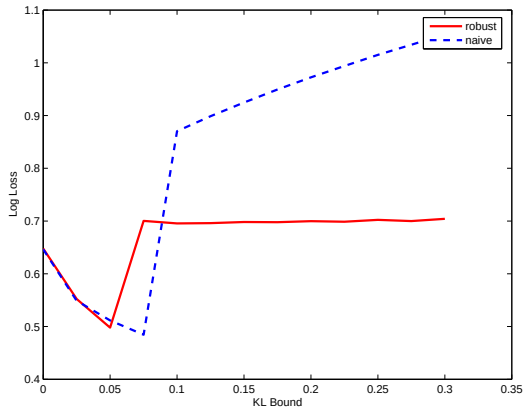


Figure 1: The graph above shows log loss on the Pima Indians dataset. We vary the KL bound for both the training and testing weighted data. The naive methods loss grows substantially faster as the KL bound on the test weights grows larger.

On typical examples we note that the test error on data that is identically distributed to the training distribution (i.e. the standard setting with unweighted test data), the error rises (relatively slowly) while we make the classifier more and more robust. This intuitively makes sense– we trade performance for robustness across more distributions. Occassionally, as in this dataset, (shown in Figure (2) left) the unweighted training error actually drops as the classifier becomes more and more robust.

Extensions

A quick derivation along the lines of that developed above will convince that the noise-free assumption and the algorithms developed in that framework of robustness can be seen instead as allowing not just the input distribution, but also the concept (i.e. $p(y|x)$) to vary. That is, suppose we attempt to solve the game:

$$\min_{\theta\in\Theta} \max_{q_{\text{test}}(x,y)\in\mathcal{Q}} \mathrm{E_q}[\ell(\mathrm{y}, \mathrm{h}_\theta(\mathrm{x}))]$$

where we take $\mathcal{Q}$ here to be the set of distributions $q(x, y)$ so that $\mathcal{KL}(q_{\text{test}}(x)|\mathcal{D}) < \epsilon$.

This has two important implications. First, it implies that our analysis and algorithms provide an additional important measure of robustness: to "concept drift" or change in $\mathcal{D}$ in response to the learner's behavior. If we are interested in only changes to $q_{\text{test}}(x)$, however, it indicates that in the noisy label case we are being overly conservative. More careful approximations should be developed for this case. In the next section we take up consideration of connections that may allow for that extension.

Relation with AdaBoost

Boosting (Schapire 1990) is a general technique for constructing a powerful learning algorithm based on a set of weak learners– that is, a single strong classifier is built from a set of classifiers that are potentially barely better than random guessing. AdaBoost (Mason *et al.* 1999) is an iterative boosting algorithm. In each round, it attempts to construct a new data-set to "weak-learn" on which the current constructed classifier appears to perform no better than random. Interestingly, AdaBoost does so by a re-weighting the training data set using an exponential of the loss function.

This is reminiscent of the scheme developed in this work. AdaBoost, of course, is solving the different problem of trying to find a single good classifier where the test and train distributions agree. Our analysis does, however, lead to some insight into the algorithm's behavior, as well as suggesting future approaches to the robust supervised learning problem. In particular, recent progress on noise-tolerant boosting (Kalai and Servedio 2003) might be folded into our attempts to build classifiers robust to changing input distributions. One interpretation of AdaBoost's potential poor noise tolerance (Mason *et al.* 1999) is that it is requiring the weak learners to perform well in the harder case where both the input distribution and concept class distributions might vary.

Future Work

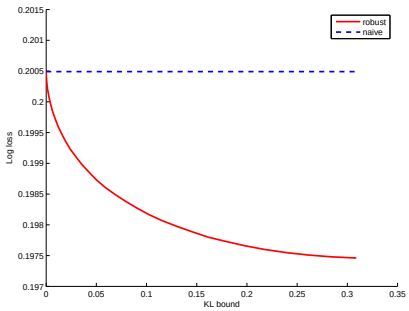
Robotics is an area where learning algorithms very much affect their own test distribution. Within the robotics learning community a class of algorithms that are locality sensitive are popular. (Moore *et al.* 1995) These models are intuitively appealing because local models do not experience "catastrophic interference" where learning in one part of an input space adversely affects things already learned in another part of the space. Our hope is that the robust framework can be extended to better understand these properties of local modeling, possibly to provide performance guarrantees and more appropriate algorithms.

We have also done initial experiments in applying our algorithm to doing generalized policy iteration with classifiers. This is an area where the classifier can dramatically effect the distribution of text instances (albeit not in a way that naturally suggests a $\mathcal{KL}$ bound). Preliminary results are encouraging in that instabilities that typically haunt generalized policy iteration seem greatly reduced.

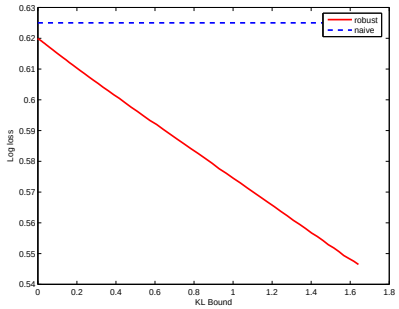
Finally, we will investigate the connections described between boosting and the robust supervised learning framework.

Acknowledgments

This work was sponsored by the Defense Advanced Research Projects Agency (DARPA) under contract "Unmanned Ground Combat Vehicle - PerceptOR Integration" (contract number MDA972-01-9-0005). The



Unweighted Loss vs. KL



Weighted Loss vs. KL

Figure 2: Iris Data Set (Left) We compare log loss on an independent test set for both the robustly trained (red solid) and standard (blue dashed) maximum entropy classifiers. In this (atypical) dataset, we gain test set improvements as the training robustness (i.e. $\mathcal{KL}$ bound) is increased. (Right) We compare Log Loss on a data set where the KL-bound on the test set is held at 0.5 while the training bound is varied. Interestingly, it is often the case that performance of the robust classifier continues to improve significantly even as the training robust passes the test bound.

views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

References

J. Bagnell and J. Schneider. Uncertain markov decision processes. Technical Report CMU-RI-TR-01-26, Carnegie Mellon University, 2001.

James Bagnell, Sham Kakade, Andrew Ng, and Jeff Schneider. Policy search by dynamic programming. In *Neural Information Processing Systems*, volume 16. MIT Press, December 2003.

D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, 2003.

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1997.

N. Daliv, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the Conference on Knowledge Discovery in Data*, 2004.

A. Fern, B. Givan, and S. Yoon. Approximate policy iteration with a policy language bias. In *Proceedings of Neural Information Processing Systems*, 2003.

David Haussler. *Part 1: Overview of the Probably Approximately Correct (PAC) Learning Framework*. 1995.

A. Kalai and R. Servedio. Boosting in the presence of noise. In *In Proceedings of the Thirty-Fifth Annual ACM Symposium on the Theory of Computing*, 2003.

J. Langford. Practical prediction theory for classification. *to appear in the Journal Machine of Machine Learning Research*, 2005.

Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. In *IEEE Symposium on Foundations of Computer Science*, 1989.

L. Mason, J.Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*. MIT Press, 1999.

A. Moore, C. Atkeson, and S. Schaal. Locally weighted learning for control. *AI Review*, 1995.

K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification, 1999.

A. Nilim and L. el Ghaoui. Robust solutions to markov decision problems with uncertain transition matrices. In *Neural Information Processings Systems 16*, 2004.

D. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems 1*, 1989.

R. E. Schapire. The strength of weak learnability. *Machine Learning*, pages 197–227, 1990.

S. Thrun. Learning to play the game of chess. In *Advances in Neural Information Processing Systems 7*. The MIT Press, 1995.