A New Boosting Algorithm Based on Dual Averaging Scheme

Nan Wang

Abstract*— The fields of machine learning and mathematical optimization increasingly intertwined. The special topic on supervised learning and convex optimization examines this interplay. The training part of most supervised learning algorithms can usually be reduced to an optimization problem that minimizes a loss between model predictions and training data. While most optimization techniques focus on accuracy and speed of convergence, the qualities of good optimization algorithm from the machine learning perspective can be quite different since machine learning is more than fitting the data. Better optimization algorithms that minimize the training loss can possibly give very poor generalization performance. In this paper, we examine a particular kind of machine learning algorithm, boosting, whose training process can be viewed as functional coordinate descent on the exponential loss. We study the relation between optimization techniques and machine learning by implementing a new boosting algorithm. DABoost, based on dual-averaging scheme and study its generalization performance. We show that DABoost, although slower in reducing the training error, in general enjoys a better generalization error than AdaBoost.

Index Terms*— Machine Learning, Optimization, Boosting, AdaBoost, computational learning theory

I. INTRODUCTION

Optimization formulations and methods lie at the heart of many machine learning algorithms [19]. A larger number of machine learning algorithms reduce to optimization problems. For example, support vector machine (SVM) [10, 22, 23] minimizes the hinge loss function between the training data and the model prediction. Hidden temporal model for sequential data [21, 2, 14, 13] maximize the conditional likelihood of observed data. Logistic regression minimized the negative log conditional likelihood of training data given the model. Reinforcement Learning problems [3, 16, 11, 12] can be formulated in terms of maximizing the sum of future rewards. Optimization algorithms are widely used for training a machine learning model. However, machine learning is more than simply a consumer of optimization techniques since machine learning concerns not only about model training but also model validation. The criterion used to validate the efficacy of a model is not the same as the criterion used for training the model. In an optimization problem, the quality of a good solution would be measured by its speed (convergence rate) and accuracy (objective gap). But in machine learning, the generalization performance is perhaps the most important metric of solution quality. An optimization algorithm that has a poor convergence rate may score a high generalization performance when it's applied to train a machine learning problem.

Therefore machine learning presents new challenges to mathematical optimization. It is still an open question on what are the desirable properties of an optimization algorithm from the machine learning perspective. In this paper, we study boosting [8, 7], a machine learning method that is famous for its resistance to over-fitting. For example, the winners of the HiggsML Challenge on Kaggle, develop and use the Boosting library, XGBoost [5], to win this competition. We formulate boosting as an optimization problem on the exponential loss and show its equivalency to gradient descent. We introduce a novel variant of boosting algorithm, based on a new optimization algorithm called the dual averaging methods that minimizes the same exponential loss function. We examine the performance of the standard boosting algorithm and ours, from both an optimization perspective and a machine learning perspective. Boosting is a general method to derive strong learner from weak learning algorithms. The boosting method is based on the observation that finding base (weak) learners that performs just slightly better than random guessing can be a lot easier than finding a single, highly accurate one. Levin et.al. [15] first postulated the conjecture of whether a combination of base learners can be boosted into an arbitrary accurate strong learner in the framework of PAC(probably approximately correct) learning model. Freund et.al. [6] introduced the first practical boosting algorithm in binary classification, called AdaBoost, which repeatedly calls a base learning algorithm to train different classifiers that fits the re-sampled training examples from a different distribution. At each round the AdaBoost algorithm assigns larger weights on the harder examples, this effectively forces the base learning algorithm to focus its attention on the examples that were misclassified by the preceding classifier, and to come up with a new classifier that is hopefully more accurate. AdaBoost then combines those weak classifiers by simply taking a weighted majority vote of their predictions. AdaBoost is shown to give significant accuracy improvements over base learning algorithms. Looking to extend upon the success of AdaBoost, many attempts have been successfully at providing general algorithms for boosting. Breiman [4] and Mason et.al. [18] made the crucial links between AdaBoost and optimization by reformulating AdaBoost from a gradient descent point of view on an exponential loss function. This intuitive connection was further developed and many variants of AdaBoost were seen as performing gradient descent, but with different loss functions and different gradient descent methods. Furthermore, as pointed out by [9] in their recent work, the existing gradient- based boosting algorithm can fail to converge on some non-smooth convex objectives.

Revised Version Manuscript Received on August 04, 2015.

Nan Wang, Linkedin Corporation, Mountain View, California, United States, America.

To address this issue, they presented new algorithms that can be extended to arbitrary convex loss functions with convergence guarantee. How- ever, one limitation of these existing algorithms is that they computed new classifier based only on the sub-gradient of loss function at previous iteration. It was known that [24] this kind gradient descent method lacks the capability in exploiting the feasible set, especially when the loss function has additional regularization term such as l_1 norm for promoting sparsity. In this project, we would like to apply gradient descent method that involves the running average of all past sub-gradients of loss function (known as dual averaging method [20, 1]), to the boosting framework. In addition, we would like to study the convergence results of the proposed algorithm. Finally, we will demonstrate experimental results that support our analysis and examples that show the need for the new algorithm based on dual-averaging scheme. The remainder of this paper is organized in the following way. Section 2 first describe the Adaboost algorithm and formulated it as a gradient descent on the exponential loss. In section 3 we introduce the dual averaging method and show how to implement it in the boosting setting. We compare the performance results of both boosting algorithms in section 4. We conclude with discussions in section 5.

II. RELATED WORK: ADABOOST

A. Algorithm description

The AdaBoost algorithm (shown in Algorithm 1) is arguably one of the most crucial developments in machine learning in the past two decades. AdaBoost can train classifiers with extreme small generalization errors from base learners as weak as decision stumps or as strong as neural networks. Let $\{(xi, yi)\}_{i=1,\ldots,n}$ be the training set where

Algorithm 1 AdaBoost

Initial $D_1(i) = \mathbf{1}/n \; \forall i = 1, \ldots, n.$
for $t = 1, \ldots, T$ **do**
  Train the weak classifier h_t with smallest training error ϵ_t with respect to D_t.
  Choose $\eta_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  Let $Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$ be a normalization factor so that D_{t+1} will be a distribution,
  Update $D_{t+1}(i) = D_t(i) \exp(-\eta_t y_i h_t(x_i))/Z_t$, for all i
  The final classifier $f_t = f_{t-1} + \eta_t h_t = \sum_{s=1}^{t} \eta_s h_s$
end for

the training instance $x_i \in X$ and the training label $y_i \in \{-1, 1\}$.

AdaBoost calls a given weak or base learning algorithm repeatedly in a series of rounds $t = 1, \ldots, T$. Adaboost maintain a distribution D_t over the training set, where $D_t(i)$ represents the weight of this distribution on training example i on round t. Initially the distribution is uniform and all weights are set equally. But on each round, AdaBoost increase the weights of incorrectly classified examples by the previous classifier and decrease the weights of correctly classified ones. In this way, the weak learning algorithm is forced to focus on the hard examples in the training set. The job of the weak learner is to find a classifier h_t that minimized the training error E_t with respect to distribution D_t:

$$\epsilon_t = \mathrm{E}_{i \sim D_t}[y_t \neq h_t(x_i)] = \sum_{i:h(x_i) \neq y_i} D_t(i) \qquad (1)$$

In practice, the weak learner may be an algorithm that can make use of the weights D_t on the training examples, or a subset of the training examples that are re-sampled according to D_t. Once the weak classifier has been trained, AdaBoost chooses a parameter $\eta_t /2 \log(1 - \varepsilon_t)/\varepsilon_t$ measures the importance that is assigned to h_t. Note that $\eta_t \geq 0$ if $E_t \leq 1/2$, and the smaller E_t gets the larger η_t becomes. The final classifier fT is a weighted majority vote of T weak classifiers where η_t is the weight assigned to h_t.

B. A gradient descent View

Here we describe the general boosting algorithm as gradient descent in function space. We consider the function $f : X \rightarrow \{-1, 1\}$ in the function space $L^2(X, \mu)$ whose Lebesgue integral is finite. The domain X is measurable and μ is a probability measure P with empirical probability distribution estimated from training instances $\{x_i\}_{i=1,\ldots,n}$. The inner product in this Hilbert space can be written as:

$$\langle f, g \rangle_{\hat{P}} = \frac{1}{n} \sum_{i=1}^{n} \langle f(x_i), g(x_i) \rangle. \qquad (2)$$

This definition of f represents a great variety of machine learning algorithms ranged from multi-layer perceptron to decision tree. Under the framework of empirical risk minimization, we would like to employ the gradient descent algorithm to minimize the empirical risk of f, which is a functional $R_{emp} : L^2 \rightarrow R$:

$$\mathcal{R}_{\mathrm{emp}}[f] = \frac{1}{n} \sum_{i=1}^{n} l(f(x_n), y_n). \qquad (3)$$

where l is the loss function that measures the difference between the prediction $f(x_n)$ and true label y_n. The gradient of R_{emp} with respect to a function f is another function g that makes $R_{emp}[f + \eta_g]$ change the most rapidly:

$$g(x) = \nabla \mathcal{R}_{\mathrm{emp}}[f](x) = \frac{\partial \mathcal{R}_{\mathrm{emp}}[F + \eta 1_x]}{\partial \eta} \mid \eta = 0 \quad (4)$$

where 1_X is the indicator function of x. In contrast to the standard gradient descent algorithm, boosting restrict a set of allowable descent directions called the feasible set H, which correspond directly to the set of hypotheses generated by the base learner. H can be a set of all possible decision trees generated by C4.5 algorithm, or a set of all possible support vector machines. Given H, we would like to find a hypothesis $h^\star$ that is the closest to the computed negative gradient. $h^\star$ can then be found by projecting the negative gradient onto H.

$$h^\star = \arg\max_{h \in \mathcal{H}} \langle -\nabla \mathcal{R}_{\mathrm{emp}}[f], h \rangle_{\hat{P}}. \qquad (5)$$

Finally the gradient descent algorithm will chose a step size η_t such that the empirical risk at the updated function $R_{emp}[f + \eta_t h^\star]$ is minimized.

Algorithm 2 Gradient Projection Algorithm

Given starting point f_0.
for $t = 1, \ldots, T$. **do**
Compute the gradient $\nabla \mathcal{R}_{\text{emp}}[f_{t-1}]$
Find $h^* = \arg\max_{h \in \mathcal{H}} \langle -\nabla \mathcal{R}_{\text{emp}}[f_{t-1}], h \rangle_{\hat{P}}$.
Find a step size $\eta_t = \arg\min_{\eta} \mathcal{R}_{\text{emp}}[f_{t-1} + \eta h^\star]$
Update $f_t = f_{t-1} + \eta_t h^\star$
end for

For Adaboost, it can be shown that $R_{emp}[f] = \sum_i \exp(-f(x_i) y_i)$. We have

$$\nabla \mathcal{R}_{\text{emp}}[f](x) = -y_i \exp(-y_i f(x_i)) \delta_{x,x_i} \qquad (6)$$

where $\delta_{x,x_i} = 1$ if $x = x_i$, otherwise $\delta_{x,x_i} = 0$. Finding the closet hypothesis $h^\star$ from H is equivalent to maximizing

$$\begin{aligned} \langle -\nabla \mathcal{R}_{\text{emp}}[f_{t-1}], h \rangle_{\hat{P}} &= -\frac{1}{n} \sum_i -y_i h(x_i) e^{(-y_i f_{t-1}(x_i))} \\ &= \sum_i y_i h(x_i) D_t(i) [\prod_{s=1}^{t-1} Z_s] \\ &\propto [1 - 2\epsilon_t] \end{aligned}$$

where $D_1(i) = \frac{1}{n}$, $\epsilon_t = \mathrm{E}_{i \sim D_t}[y_t \neq h_t(x_i)]$ and

$$\begin{aligned} D_t(i) &= \frac{\exp(-y_i f_{t-1}(x_i))}{n \prod_{s=1}^{t-1} Z_s} \\ &= \frac{D_{t-1}(i)}{Z_{t-1}} \exp(-\eta_{t-1} y_i h^\star(x_i)) \qquad (7) \\ Z_t &= 2\sqrt{\epsilon_t(1-\epsilon_t)} \qquad (8) \end{aligned}$$

This projection step is equivalent to finding a base hypothesis with smallest misclassification error ε_t over a boosted training set with distribution D_t. Finally, we choose the step size η_t such that

$$\begin{aligned} &\frac{d\mathcal{R}_{\text{emp}}[f_{t-1} + \eta h]}{d\eta} \\ &= -\sum_{i=1}^{n} y_i h(x_i) \exp(-y_i f_{t-1}(x_i) - y_i h(x_i) \eta) \\ &\propto -\sum_{i=1}^{n} y_i h(x_i) D_t(i) \exp(-y_i h(x_i) \eta) \\ &= \epsilon_t \exp(\eta) - (1 - \epsilon_t) \exp(\eta) = 0 \qquad (9) \\ \eta_t &= \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} \qquad (10) \end{aligned}$$

Therefore, with exponential loss function, the gradient projection algorithm 2 is equivalent to the AdaBoost algorithm 1. By viewing AdaBoost as a gradient descent in the functional space, it's tempting to conclude that AdaBoost a just an algorithm for minimizing exponential loss and more (less) powerful optimization techniques for the same loss should work even better (worse). In the next sections, we are going to test this conclusion by introducing a new variant of boosting algorithms that implements a different optimization technique.

III. Boosting with Dual Average Method

Dual averaging method (shown in algorithm 3) has recently been introduced in convex optimization by [20]. In the paper of [1], they proposed an alternative viewpoint of the Hedge algorithm using dual averaging method. The hedge algorithm has been known for its close relation to the AdaBoost algorithm. However, the hedge algorithm and the AdaBoost algorithm differ in many different ways. First, in hedge algorithm (see [1] for more details), the weight D_t (i) increases if ith strategy is a "good" action at round t, while in AdaBoost the weight D_t (i) increases if the t hypothesis suggests a "bad" prediction on the ith example. The loss in hedge algorithm that measures the success of the strategy is a actually a measurement of hardness of an example in AdaBoost. Thus, the algorithm that minimizes the loss in Hedge algorithm will not minimized the training error E_t in AdaBoost. Secondly yet more importantly, in AdaBoost the updating rule for the weights is $D_{t+1}(i) \propto D_t(i) \exp(-\eta_t)^{y_i h(x_i)}$ with a time-varying parameter $\exp(-\eta_t)$ that changes at each iteration according to training error E_t. But in hedge algorithm, this parameter is fixed ahead of time.[1] Therefore, the algorithm described in [1] can not directly applied to a boosting algorithm. Instead, we need to design a novel dual averaging algorithm that is based on boosting setting.

$$\begin{aligned} h^\star &= \arg\min_{h \in \mathcal{H}} \sum_i \frac{y_i h(x_i)}{m} \sum_{k=1}^{t-1} \lambda_k \exp[-y_i f_k(x_i)] \\ &= \arg\min_{h \in \mathcal{H}} \sum_i \frac{y_i h(x_i)}{m} D_t(i) \\ &= \arg\min_{h} E_{i \sim D_t}[1 - 2\epsilon_t] \qquad (12) \end{aligned}$$

where $\epsilon_t = \mathrm{E}_{i \sim D_t}[y_t \neq h_t(x_i)]$ the probability (weight) for each instance

$$D_t(i) = \frac{\sum_{k=1}^{t-1} \lambda_k \exp[-y_i f_k(x_i)]}{\sum_{i=1}^{n} \sum_{k=1}^{t-1} \lambda_k \exp[-y_i f_k(x_i)]} \qquad (13)$$

To correctly apply the dual averaging method in the boosting setting, we need to define the dual variable $s_t = \sum_{k=1}^{t} \lambda_k g_k$ in the functional space:

$$s_t = \sum_{k=1}^{t} -\lambda_k y_i \exp(-y_i f(x_i)) \delta_{x,x_i}. \qquad (11)$$

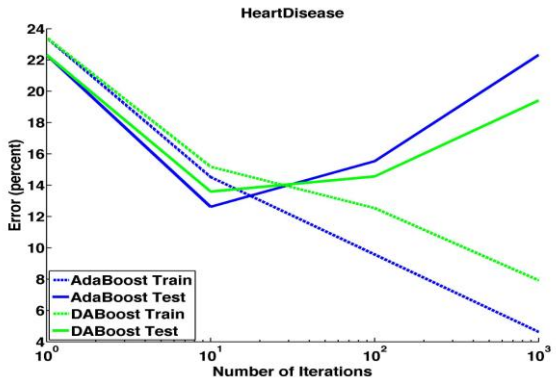
Because the hypothesis class H applies to arbitrary weak learner. It's not clear how to define a regularization function of rule-based learner (like zero-R) and tree-based learner (such as decision stumps and CART). We let $R(h) = 0$, then find a weak classifier h^* can be written as:

Algorithm 4 DABoost

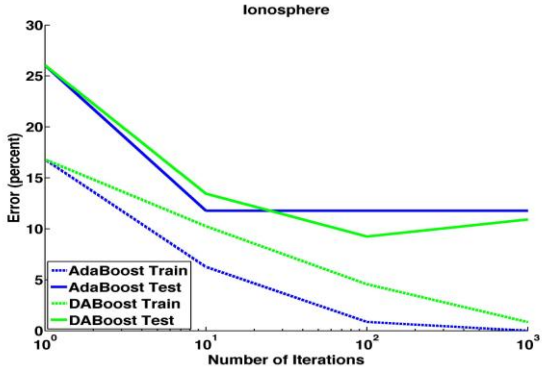
Given training samples $(x_1, y_1), \ldots, (x_n, y_n)$
Initialize $D_1 = \frac{1}{n}$
for $t = 1, \ldots, T$ **do**
  Re-sample training data from D_t.
  Find the weak classifier h_t that minimize the error rate ϵ_t.
  Set the step size $\alpha_t = \frac{1}{2}\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$
  Update distribution $D_{t+1} \propto \sum_k \lambda_k \exp[-y_i f_k(x_i)]$
end for
Final classifier $f(x) = \sum_t \alpha_t h_t(x)$

Equation 13 defines a way to update the distribution D_t in dual averaging setting. And we choose the time step η_t as in equation 9, $\eta_t = 0.5 \log((1 - \varepsilon_t)/\varepsilon_t)$. Finally we introduce a novel boosting algorithm, called DABoost (shown in algorithm 4), based on dual averaging method. Based on algorithm 4, we implement the DABoost algorithm in the WEKA environment so that our DABoost al- gorithm can call basically any existing machine learning algorithms as the base learner. We fix the time-dependent importance parameter $\lambda_t = 1$ be constant in the implementation.

IV. RESULTS



(a)



(b)

Figure 1: This figure shows the Training and test errors of AdaBoost and DABoost, both using stumps as base learner, on data sets (a)Heart Disease, (b)Inonospear DABoost tends to produce classifiers wit higher bias but less variance.

In this section, we evaluate our DABoost algorithm using different data sets in UCI machine learning repository [17], and compare our training error and test error to those of AdaBoost. Because the training part of both AdaBoost and DABoost can be viewed as convex optimization on the exponential loss, the training error represents the objective gap of the loss function, a measurement of quality from the optimization perspective, while the test error represent the generalization performance that is a measurement of quality from the machine learning perspective. We start with a toy example where the instance *x* is drawn uniformly from $[-1, 1]^{100}$, and the label *y* is the majority vote of three coordinates. The size of the training set $n = 1,000$. We use this data set to test the correctness of our DABoost algorithm. With this simple data set, both DABoost and AdaBoost (boosting stumps) achieve 0% training error after three iterations, as expected. In comparison, popular machine learning algorithms such as SVM, logistic regression and multilayer perceptron only score test errors greater than 15% (note that tree-based algorithms such as CART and C4.5 can also score 0% test error). Figure 1(a) shows the performance of AdaBoost and DABoost, both using stumps as base learner, in Heart Disease data set. The blue curves represent the results from Adaboost while the green curves represent the results from DABoost. Training errors are shown in dashed lines while test errors are shown in solid lines. The heart disease data set contains 14 attributes. The label refers to the presents of heart disease in the patient. The heart disease data set is one of few training sets on which the AdaBoost algorithm overfits. As shown in the figure, DABoost converges slowly in terms of training error but suffers less over-fitting in terms of test error. Figure 1(b) shows the performance of AdaBoost and DABoost, both boosting decision stumps, in ionosphere data set The ionosphere data set contains 34 continuous attributes. The label is either "good" or "bad" where "good" radar returns are those showing evidence of some type of structure in the ionosphere and "bad" returns are those that do not. The ionosphere data set is commonly used in the machine learning literature. As shown in the figure, DABoost again converges slower than Adaboost in terms of training error but achieves better test error. In general, DABoost gets higher training error but enjoys lower test error. Similar behavior have been observed in many other data sets such as the letter data set by boosting a C4.5 base learner, and in the diabetes data set by boosting stumps. This is due to the bias-variance trade-off. DABoost update the distribution *Dt* based on the average of loss over all previous iterations. The resulting classifier becomes less flexible because as the number of iterations *t* increases, the distribution *Dt* changes slowly due to the update rule in equation 15. Thus, DABoost tends to produce classifiers wit higher bias but less variance. However, DABoost doesn't always have better generalization performance than Adaboost. Figure 4 shows the performance of AdaBoost and DABoost, whose base learner is decision stumps, in Webb Spam Corpus data set. Web spam is defined as Web pages that are created to manipulate search engines and deceive Web users. All positive examples were taken and the negative examples were created by randomly traversing the Internet starting at well known (e.g. news) web sites. In this data set, any continuous one byte is treated as a word and the world count is used as the feature value. Each instance is normalized to unit length.

The number of total features is 254. Due to memory constraints, only 1% of instances (3, 500 instances) are used for training. As shown in the figure, DABoost converges slower than AdaBoost in terms of training error but also suffers larger test error. The reason for such poorer performance is due to the noise in the highly biased labels in the data set.

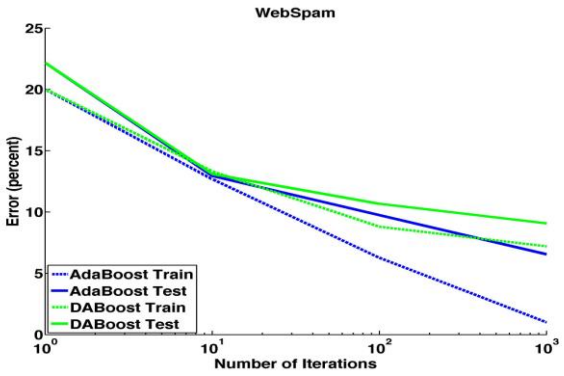


Figure 2:This figure shows the Training and test errors of AdaBoost and DABoost, both using stumps as base learner, on the data set Web spam.

V. CONCLUSION

In this paper, we discuss the quality of a good optimization algorithm from both a machine learning perspective and a mathematical programming perspective. We postulated that a slower convergent optimization algorithm might result in a better machine learning algorithm with better generalization performance. We test this postulation by introducing a new variant of boosting algorithm, DABoost, based on dual averaging gradient descent method on exponential loss. Our simulation results show although slower in obtaining small training error, DABoost in general enjoys better generalization error than AdaBoost. Our implementation of DABoost is still far from complete and demands a series of future research. We fix the time- dependent importance parameter $\lambda t = 1$ be constant in the implementation. A time-varying λt might lead to different results. Moreover, we simplifies the dual averaging algorithm by restraining the regularization function $d(h) = 0$. $l1$ or $l2$ regularization might be applied to base learners that can be parametrized by a vector of real numbers. DABoost is based on dual averaging algorithm, the recently introduced convex optimization algorithm that has similar linear convergence rate as the gradient descent. In our further work, more powerful optimization techniques such as accelerated gradient descent with quadratic convergence rate might be implemented in the boosting framework.

REFERENCES

1. M. Baes and M. Burgisser. Hedge algorithm and dual averaging schemes. arXiv, 1112(1275), 2011.
2. LR Bahl, Peter F Brown, Peter V De Souza, and Robert L Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In proc. ICASSP, volume 86, pages 49–52, 1986.
3. Craig Boutilier. A POMDP formulation of preference elicitation problems. In AAAI/IAAI, pages 239–246, 2002.
4. L. Breiman. Prediciton games and acring algorithms. Nueral Computation, 11:1493–1517, 1999.
5. Tianqi Chen, Hang Li, Qiang Yang, and Yong Yu. General functional matrix factorization using gradient boost- ing. In Proceeding of 30th International Conference on Machine Learning (ICML'13), volume 1, pages 436–444, 2013.
6. Y. Freund and R. E. Schapire. A decision-theorectic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
7. Yoav Freund, Robert Schapire, and N Abe. A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780):1612, 1999.
8. Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1): 119–139, 1997.
9. A. Grubb and J. A. Bagnell. Generalized boosting algorithms for convex optimization. In Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, June 2011.
10. Marti A. Hearst, Susan T Dumais, Edgar Osman, John Platt, and Bernhard Scholkopf. Support vector machines. Intelligent Systems and their Applications, IEEE, 13(4):18–28, 1998.
11. Y. Huang, A. L. Friesen, T. D. Hanks, M. N. Shadlen, and R. P. N. Rao. How prior probability influences decision making: A unifying probabilistic model. Advances in Neural Information Processing Systems (NIPS), 2012.
12. Y. Huang and R. P. N. Rao. Reward optimization in primate brain: A POMDP model of decision making under uncertainty. PLoS One, 8(1), 2013.
13. Yanping Huang and Rajesh P Rao. Neurons as monte carlo samplers: Bayesian inference and learning in spiking networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 1943–1951. Curran Associates, Inc., 2014.
14. Yanping Huang and Rajesh PN Rao. Predictive coding. Wiley Interdisciplinary Reviews: Cognitive Science, 2(5): 580–593, 2011.
15. M. J. Kearns and L. G. Valiant. Learning boolean formulae or finte automata is as hard as factoring. Technical report, Department of Computer Science, Harvard University, 1988.
16. Esther Levin, Roberto Pieraccini, and Wieland Eckert. Using markov decision process for learning dialogue strategies. In Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on, volume 1, pages 201–204. IEEE, 1998.
17. M. Lichman. UCI machine learning repository, 2013.
18. L. Mason, J. Baxter, P. Barlett, and M. Frean. Functional gradient techniques for combining hypotheses. In Advances in large margin classifiers. MIT Press, Cambridge, 1999.
19. Kevin P Murphy. Machine learning: a probabilistic perspective. MIT press, 2012.
20. Y. Nesterov. Primal-dual subgradient methods for convex problems. Mathematical Programming, 120(1): 221– 259, 2009.
21. Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. ASSP Magazine, IEEE, 3(1): 4–16, 1986.
22. Bernhard Scho¨lkopf and Alex Smola. Support vector machines. Encyclopedia of Biostatistics, 1998.
23. Ingo Steinwart and Andreas Christmann. Support vector machines. Springer Science & Business Media, 2008.
24. L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. The Journal of machine learning research, 11, 2010.