ENSEMBLE CLASSIFICATION BASED ON RANDOM LINEAR BASE CLASSIFIERS

Qi Xiao and Zhengdao Wang

Department of Electrical and Computer Engineering, Iowa State University, USA

ABSTRACT

We propose a simple ensemble classification algorithm, which employs a set of N randomly generated linear classifiers, followed by a selection process based on the performance of these classifiers on the whole set of training data. The top n performers are then linearly combined to form the final classifier. We analyze the VC dimension of the resulting hypothesis set from such a construction procedure, and show that it can be controlled by choosing the parameters N and n. The proposed algorithm enjoys low computational complexity, and for the MNIST dataset and several UCI datasets that we tested, the algorithm compares favorably in generalization error rate or running time to competing algorithms including Random Kitchen Sinks and AdaBoost.

Index Terms— Ensemble learning, AdaBoost, Random Kitchen Sinks, randomization

1. INTRODUCTION

Linear model is a basic and important model in classification. To classify the data linearly, the data points are given numerical labels. The predicted label is modeled as a linear function of data features. The task is to find a linear function that fits the training data according to certain optimality criterion. The single best function, even if it can be identified, might not perform well outside the training data due to overfitting. It is well known that the VC dimension for a linear classifier is $d+1$, where d is the feature vector dimension [1]. So for large d, overfitting is likely. If we add some constraints on the linear classifier, the hypothesis set space becomes smaller, which causes the VC dimension to be smaller. For example, [2] gives a bound on VC dimension of sparse linear classifier.

For data that are not linearly separable, nonlinear models may be used. The model used can be highly complex with many parameters, which is the case in deep learning networks. Even though deep learning networks have been shown to work well empirically, there is current a lack of solid theoretical understanding of why it works well.

The ensemble approach is another way of building a strong learner for complex machine learning problems [3, 4, 5]. The idea is to build a stronger learner by combining a collection of many weak base learners. An ensemble learning method consists of two components: the set of base learners and the combining rule used to construct the ensemble learner, e.g., [6]. The base learners in a good ensemble learner have two properties. The first one is that individual base learners have good performance on training data (better than random guess), and the second one is that they are different from each other so that sufficient diversity exist in the ensemble [7].

To generate diverse base learners, a typical way is to make use of randomness. The method of bagging [8] samples the training data with replacement multiple times to create a collection of training sets. A base learner is then trained on these training sets and their outputs are averaged for regression, or majority voted for classification. Another idea is to randomly choose a subset of data to train a weak learner during each iteration. Random projection [9, 10] can also be used such that the base learner is trained on different random projections of the given training data.

A large class of boosting algorithms use iterative search to refine the ensemble classifier by reweighing the samples adaptively, including AdaBoost [11] and its variants [12]. Such boosting algorithms have been shown to be successful in solving a large number of learning problems. It has been observed that they rarely suffer from the overfitting problem, even when a large number iterations are performed. There have been several theoretical analyses of the algorithm from different perspectives, including margins, statistical interpretation and coordinate descent, game theoretic interpretation, and information geometry; see [12]. Despite the progress, there is still a need to theoretically understand why AdaBoost is robust to overfitting. Also the problem of AdaBoost's sensitivity to noise needs to be addressed.

Due to the fact that randomization is computationally cheaper than optimization, an algorithm called Random Kitchen Sinks [13] was put forward. Unlike AdaBoost, which jointly minimizes base learners parameters and their coefficients in the final ensemble, the algorithm first randomly generates the parameters of base learners, then optimizes their coefficients. Simulations show that this algorithm can achieve the same accuracy as AdaBoost at a faster speed.

In this paper we propose two simple ensemble classification methods that involve a set of randomly generated linear classifiers. These linear classifiers are fixed and do not need to be trained. Our combining rule consists of two steps: first the set of fixed base classifiers will be ranked according to their error rates. In the first proposed method, a certain number of top performers will be then combined with equal weights. The resulting ensemble classifier is such that for a testing sample, it will perform majority voting among the top performers of the base linear classifiers. The second method applies weighted majority voting among the same top performers by optimizing the combining coefficients. The proposed classifiers are simple and computationally fast. The generalization error performance can be quantified through VC dimension analysis thanks to its simplicity. We provide such an analysis in the paper. The complexity and performance of the proposed method can be adjusted by adjusting the few parameters. We have compared our methods with competing algorithms including Random Kitchen Sinks and AdaBoost using numerical simulations. The results are favorable in terms of generalization error and comparable in terms of running time.

2. PRELIMINARIES

Assume that the input $x \in \mathbb{R}^d$ and output $y \in \{+1, -1\}$ follow a fixed but unknown distribution $P(x, y)$. We would like to learn

the dependency between x and y based on a set of m samples: $\{(x_i, y_i)|i = 1, ..., m\}$, generated independently and identically (iid) from a distribution P. We augment each x_i by appending a constant 1 at the end such that $x_i \in \mathcal{X} = \mathbb{R}^{d+1}$. This augmentation facilitates expressing a possible shift of a hyperplane in $\mathbb{R}^d$.

A learner aims at constructing a classification function $\hat{f} : \mathcal{X} \to \{+1, -1\}$ which can be used to predict the label y based on x. The empirical risk measures how $\hat{f}$ fits the training data,

$$E_{\text{emp}}[\hat{f}] \equiv \frac{1}{m} \sum_{i=1}^{m} c(\hat{f}(x_i), y_i). \tag{1}$$

The loss function $c(\cdot, \cdot)$ evaluates the inconsistency between predicted label $\hat{f}(x)$ and the true label y. Common choices for c are 0-1 loss, $I(\hat{f}(x) \neq y)$, hinge loss, $\max(0, 1 - \hat{f}(x)y)$, exponential loss, $e^{-y\hat{f}(x)}$, and the quadratic loss, $(y - \hat{f}(x))^2$.

The empirical risk is an approximation to the true risk function. For example, the true risk function for empirical risk with 0-1 loss function is,

$$E[\hat{f}] \equiv \Pr_{(x,y)\sim P}(\hat{f}(x) \neq y). \tag{2}$$

A *linear classifier* is such that the hypothesis function $\hat{f}$ takes a linear form:

$$\hat{f}_w(x) = \text{sign}(w^T x) \tag{3}$$

where $w \in \mathbb{R}^{d+1}$ is the weight vector of the linear classifier. It is possible to construct nonlinear classifiers based on linear ones, by first mapping the data nonlinearly to some feature space. The classification function has form $\hat{f}(x) = \text{sign}(\sum_{i=1}^{\infty} \alpha(w_i)\phi(x; w_i))$, where feature functions $\phi : \mathcal{X} \times \Omega \to \mathbb{R}$ parameterized by $w \in \Omega$, are weighted by coefficients $\alpha \in \mathcal{A}$. AdaBoost [11] greedily minimize over α, w jointly by solving the following optimization problem:

$$\underset{w_1,\ldots,w_n \in \Omega, \alpha \in \mathcal{A}}{\text{minimize}} \quad E_{\text{emp}} \left[\sum_{j=1}^{n} \phi(x; w_j)\alpha_j \right]. \tag{4}$$

Random Kitchen Sinks [13] on the other hand, first draws the parameters of the nonlinearities $\{w_j\}_{j=1}^n$ randomly. Given fixed w, it fits the coefficients $\alpha := [\alpha_1, \ldots, \alpha_n]$ via convex optimization within the set

$$\mathcal{A} = \{\alpha | \, \|\alpha\|_\infty \leq C/n\}, \tag{5}$$

where C is a scalar, and $\|\cdot\|_\infty$ denotes the ∞-norm.

3. PROPOSED ALGORITHM

Motivated by ensemble learning and a desire to reduce computational complexity of ensemble classifiers and increase robustness against noise, we propose a simple ensemble classifier in this section. We choose our base classifiers to be linear classifier in form of $\text{sign}(w^T x)$. We then combine the base classifiers for prediction. In the next section we will analyze the performance of our proposed classifier in terms of VC dimension and the generalization error.

We randomly generate a pool of N weights as our base classifiers. We then rank them based on their performance in terms of error rate when they are applied individually to the training data set. We choose a small number n of the top performers and combine them using equal or unequal weights. The combined classifier is the final ensemble classifier.

Algorithm 1 Random Ensemble algorithm

Input: m training data points $\{(x_i, y_i)|i = 1, ..., m\}$, integer N, n, a probability distribution $p(w)$

Output: A classifier $\hat{f}(x) = \text{sign}\left(\sum_{j=1}^{n} \text{sign}(\tilde{w}_j^T x)\right)$

1) Generate $w_1, w_2, ..., w_N$ iid from $p(w)$.
2) Calculate the error rate of each classifier w_j as

$$E_j = \frac{1}{m} \sum_{i=1}^{m} I(\, \text{sign}(w_j^T x_i) - y_i \,), \quad j \in [1, N] \tag{6}$$

where $I(x) = 1$ if $x \neq 0$ and $I(0) = 0$.
3) Rank the N error rates $\{E_j | 1 \leq j \leq N\}$ in increasing order. Let the first n weights denoted as $\{\tilde{w}_j | 1 \leq j \leq n\}$.
4) The output ensemble classifier is obtained by simple majority voting (7), or via the weighted combining (8) and (9).

Two proposed ways of combining the top performers are:
1) *Equal weight combining:* In this case, the final ensemble classifier is given as

$$\hat{f}(x) = \text{sign}\left(\sum_{j=1}^{n} \text{sign}(\tilde{w}_j^T x) \right). \tag{7}$$

2) *Optimized weight combining:* As well as majority voting combination rule in Algorithm 1, we consider weighted majority combining after we obtain the top n classifiers. In order to get the optimal coefficients for the weighted majority voting scheme, we solve the ridge regression problem as Random Kitchen Sinks did in their experiments [13],

$$\underset{\alpha \in \mathbb{R}^n}{\text{minimize}} \quad \frac{1}{m} \sum_{i=1}^{m} \left[\sum_{j=1}^{n} \alpha_j \, \text{sign}(\tilde{w}_j^T x_i) - y_i \right]^2 + n\lambda \, \|\alpha\|_2^2 \tag{8}$$

where λ is a scalar related to C in [13]. By replacing infinity norm in (5) with quadratic norm, we get $\mathcal{A}' = \{\alpha | \, \|\alpha\|_2^2 \leq C^2/n\}$. Thus $n\lambda$ is the coefficient of second term in our objective function. We use $\phi(w; x) = \text{sign}(w^T x)$ to extract nonlinear feature.

After obtaining the optimal solution $\alpha^\star$, the final ensemble classifier based on weighted majority voting is

$$\hat{f}_{\text{opt}}(x) = \text{sign}\left(\sum_{j=1}^{n} \alpha_j^\star \, \text{sign}(\tilde{w}_j^T x) \right). \tag{9}$$

We remark that the proposed algorithm yields finally a nonlinear classifier. The final ensemble classifier should work well if the data are linearly separable and the parameters N and n are chosen appropriately. At the extreme, if $N \to \infty$ and $n = 1$, then the chosen best classifier should be close to the best possible linear classifier based on the training data set. Compared to a linear classifier that is directly optimized based on the training dataset, the proposed method is computationally simpler. Also, through the choices of the parameters N and n, the VC dimension, and hence the generalization error, can be controlled.

4. PERFORMANCE ANALYSIS

Generalization error can be measured based on in-sample (empirical) error risk E_{in} and true out-of-sample error E_{out}, e.g., [14]. The

success of generalization depends on two parts: 1) E_{out} can be tracked by E_{in}; 2) E_{in} is small. The *VC dimension* d_{vc} is define as the largest number of points that the hypothesis set H can shatter, such that all possible dichotomies of the points can be achieved by the set of hypotheses. When the VC dimension is finite, learning is possible in the probably and approximate correct (PAC) sense, e.g., [15]: for any tolerance $\delta > 0$, and for all $h \in H$,

$$E_{\text{out}}(h) \leq E_{\text{in}}(h) + \sqrt{\frac{8}{m} \ln \frac{4m_H(2m)}{\delta}} \tag{10}$$

with probability larger than $1-\delta$. It can be seen that as m increases, the bound gets tight, which means learning is possible.

4.1. Connection with Computation Network

Our final decision function $\hat{f}$ consists of n base classifiers $\text{sign}(\tilde{w}_i^T x)$ with $1 \leq i \leq n$. We analyze the unweighted case first, then we extend to the weighted one. We can express the classifier in terms of the original set of N randomly generated classifiers as follows

$$\hat{f}(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i \, \text{sign}\left(w_i^T x\right)\right) \tag{11}$$

where α_i is either zero or one, depending on whether the linear classifier w_i ranked among the best n ones for the training dataset. Viewed in this way, our combining rule is a linear combination with a sparsity constraint such that no more than n combining coefficients are non-zero. Actually since all of our combining coefficients are 1, the combining rule is a very specific sparse combining rule.

Based on the above discussion, our decision making process can be viewed as a two-layer feedforward network, in the flavor of [16]. An upper bound on the growth function of the decision hypothesis set based on VC dimension of each computational node was provided in [16]. In our case, there are N nodes in the first layer, where each node is a function $\hat{f}_i(x) = \text{sign}(w_i^T x)$ having randomly generated weights. The second layer only contains one node $\hat{f}$ and it connects to every node in the first layer.

Applying the analysis result of [16], for the computation nodes in first layer, each independently refers to linear threshold classifier set $H_1 = \{\text{sign}(w^T x) | w \in \mathbb{R}^{d+1}\}$, the VC dimension for set H_1 is $d+1$ [1]. Let d_{vc} denote the VC dimension of the single second layer node, then the sum of VC dimensions over all computation nodes is $N(d+1) + d_{\text{vc}}$. Thus the number of different functions that can be realized by $\hat{f}$ when the domain is restricted to a set of size m is at most

$$\left(\frac{em(N+1)}{N(d+1)+d_{\text{vc}}}\right)^{N(d+1)+d_{\text{vc}}} \tag{12}$$

where e is the base of natural logarithm. The growth function can then be used to bound the generalization error.

4.2. VC dimension of the proposed hypothesis set

The fact that the VC dimension can be as large as $N(d+1) + d_{\text{vc}}$ is rather pessimistic because N may be large. So the above analysis does not offer us a promising bound on the generalization error. The reason for the pessimistic result is that the analysis assumed that the first layer nodes are individual linear learners that have adjustable weights. In our case the weights are generated (randomly) once and then fixed. Therefore the number of hypotheses that can be realized by our scheme is actually much smaller. In particular, we have the following result on the VC dimension of our proposed classifier.

Theorem 1. *For a fixed set of N linear classifiers. The linear classifier* (7) *generated by the process in Algorithm 1 belongs to a hypothesis set whose VC dimension d_{vc} is upper bounded by* $\log_2 \binom{N}{n}$.

Proof. The classifier can be written in the form of (11). We can project the data to a n-dimensional subspace corresponding to the n nonzero α_j coefficients. After projection, the number of dichotomies that can be generated is only 1, since the coefficients for the coordinates within the subspace are fixed to be all 1. There are $\binom{N}{n}$ such projections, thus the total number of dichotomies is at most $\binom{N}{n}$. The VC dimension is therefore upper bounded as by $\log_2 \binom{N}{n}$. □

We remark that as opposed to $(d+1)$, which is the VC dimension of a linear classification scheme, our proposed ensemble classifier has a VC dimension that is independent of the dimensionality of the underlying vector space d. This is a desirable feature because d in general can be quite large. Also, through the setting of the parameters N and n we can control the desired complexity of the hypothesis set quite easily.

If we fix N, VC dimension bound will be increased as n increases. Furthermore, for a fixed n, as N increases, the VC dimension can be approximated as $d_{\text{vc}} \approx n \log_2 N - \log_2(n!) \approx n \log_2 N$.

Theorem 2. *If we allow the n non-zero coefficients of α_i in* (11) *to be any real numbers, then the classifier generated by the process in Algorithm 1 belongs to a hypothesis set whose VC dimension d_{vc} is upper bounded by* $1 + n + \log_2 \binom{N}{n}$.

Proof. Since there are at most $\binom{N}{n}$ projections to the n-dimensional subspace, and for each projection, there are at most 2^{n+1} dichotomies that can be formed, the total number of dichotomies is upper bounded by $2^{n+1}\binom{N}{n}$. The VC dimension result then follows by taking the logarithm. □

5. NUMERICAL RESULTS

5.1. MNIST dataset

We implemented the algorithm in MATLAB and tested it on the MNIST database of handwritten digits [17]. We picked the 0 and 1 digit pair from the ten digits for binary classification.

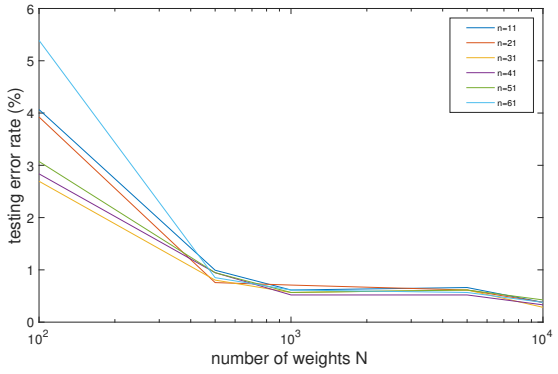

Fig. 1: Testing error rate vs. number N of weights for different n

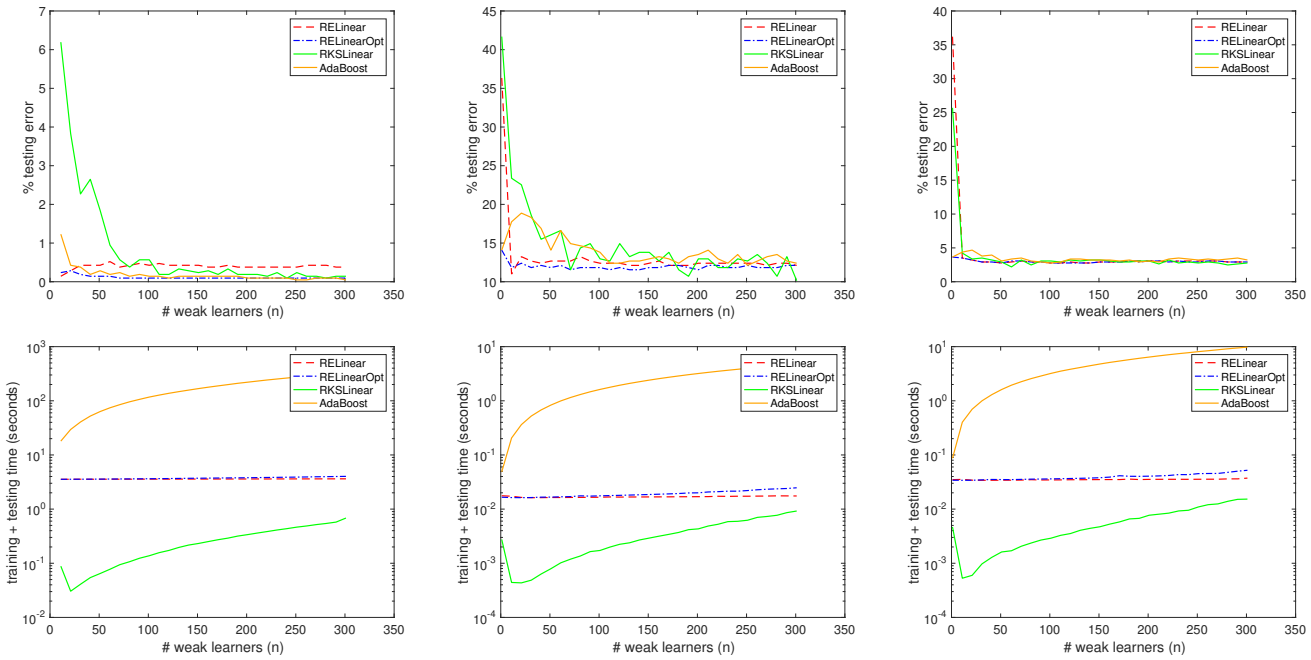

Fig. 2: Comparisons of our methods, Random Kitchen Sinks, and AdaBoost. First row: testing error versus number of weaker learners n. Second row: running (including both training and testing) time versus n. First column: Digit01 dataset. Second column: Ionosphere dataset. Third column: Breast Cancer dataset. RELinear and RELinearOpt are short names for the proposed unweighted and weighted algorithm, respectively. RKSLinear represents Random Kitchen Sinks with linear base learners.

For convenience, we refer to it as Digit01. The feature dimension of Digit01 is $d = 784$. The number of training samples is $m = 12665$. We tried (N, n) pairs where $N \in \{100, 500, 1000, 5000, 10000\}, n \in \{11, 21, 31, 41, 51, 61\}$. Figure 1 depicts the testing error rate in different (N, n) values. It reveals the decreasing tendency of error rate as N or n increases. There is diminishing return, however, as n becomes larger.

5.2. Comparisons with Competing Methods

We compared our proposed method with two other ensemble methods, namely Random Kitchen Sinks and AdaBoost. We tested our algorithm and these two algorithms on the MNIST dataset and some of the UCI datasets [18]. The datasets we used were Ionosphere and Breast Cancer. Feature dimension and number of training samples in Ionoshere are $d = 34, m = 280$. For Breast Cancer dataset, $d = 10, m = 546$. We kept $N = 10,000$ in our methods; a larger N can improve the accuracy as indicated in Digit01 results. For our weighted combining scheme and for Random Kitchen Sinks, we used quadratic loss function and quadratic constraints in convex problem (8). Parameter λ was selected in the range where $\log_2 \lambda \in \{-16, -14, ..., 8\}$. Each algorithm was tested using 5-fold cross validation. Figure 2 shows the testing error rate and running time using different numbers of weak learners n. The running time was measured on a Macbook laptop, including training and testing time.

Testing error results of the proposed two approaches are comparable to Random Kitchen Sinks and AdaBoost. Our weighted approach achieves similar or better performance than the unweighted version and Random Kitchen Sinks. This illustrates two points. First, optimizing weighted majority voting coefficients can improve the generalization performance compared to the unweighted majority voting. On the other hand, we can see that the top ranked performers offer more information about the true labels than the random ones.

In terms of running time, the proposed methods are faster than AdaBoost, but slower than Random Kitchen Sinks. Compared to these methods, the proposed methods' complexity are not sensitive to the number of weak learners. Since sorting is quite efficient, the biggest portion in the running time is calculating the N individual learners' classification errors. This can be accelerated by distributed parallel computing. Also, if we consider the time of choosing parameter λ, the proposed unweighted scheme can save lots of computational cost especially when data size is large.

6. CONCLUSIONS

We proposed an algorithm for binary classification based on a random classifier ensemble. The algorithm selects a small subset of top performing base linear classifiers from the set of randomly generated linear classifiers. A weighted or unweighted combining rule is then used to combine these top base classifiers to form the final classifier. We provided an analysis of the VC dimension for our proposed method, which is in the order of $n \log_2 N$, where N is the number of base classifiers and n is the number of selected classifiers. The VC dimension can be controlled easily through the selection of the parameters N and n. We tested our proposed algorithm on the MNIST dataset and several UCI datasets, and compared with competing algorithms including Random Kitchen Sinks and AdaBoost. The proposed methods enjoy fast training process compared with other ensemble methods, and comparable or favorable generalization performance.

7. REFERENCES

[1] Roberta S Wenocur and Richard M Dudley, "Some special vapnik-chervonenkis classes," *Discrete Mathematics*, vol. 33, no. 3, pp. 313–318, 1981.

[2] Tyler Neylon, *Sparse solutions for linear prediction problems*, Ph.D. thesis, New York University, 2006.

[3] Ludmila I Kuncheva, *Combining pattern classifiers: methods and algorithms*, John Wiley & Sons, 2004.

[4] Lior Rokach, *Pattern classification using ensemble methods*, vol. 75, World Scientific, 2009.

[5] Zhi-Hua Zhou, *Ensemble methods: foundations and algorithms*, CRC Press, 2012.

[6] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.

[7] Ludmila I Kuncheva and Christopher J Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, no. 2, pp. 181–207, 2003.

[8] Leo Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[9] Alon Schclar and Lior Rokach, "Random projection ensemble classifiers," in *Enterprise information systems*, pp. 309–316. Springer, 2009.

[10] Timothy I Cannings and Richard J Samworth, "Random projection ensemble classification," *arXiv preprint arXiv:1504.04595*, 2015.

[11] 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*, vol. 55, no. 1, pp. 119–139, 1997.

[12] Robert E Schapire and Yoav Freund, *Boosting: Foundations and algorithms*, MIT press, 2012.

[13] Ali Rahimi and Benjamin Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Advances in neural information processing systems*, 2009, pp. 1313–1320.

[14] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin, *Learning from data*, AMLBook, 2012.

[15] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar, *Foundations of machine learning*, MIT press, 2012.

[16] Eric B Baum and David Haussler, "What size net gives valid generalization?," *Neural computation*, vol. 1, no. 1, pp. 151–160, 1989.

[17] Yann LeCun, Corinna Cortes, and Christopher JC Burges, "The mnist database of handwritten digits," 1998.

[18] https://archive.ics.uci.edu/ml/datasets.html, "UCI Machine Learning Repository: Data Sets," .