

---

# Specialized Weighted Majority

## Statistical Techniques in Robotics (Fall 2009)

---

**Keywords:** classifier ensembling, online learning, expert combination, machine learning

**Javier Hernandez**  
**Alberto Rodriguez**  
**Tomas Simon**

JAVIERHE@ANDREW.CMU.EDU  
ALBERTOR@ANDREW.CMU.EDU  
TSIMON@ANDREW.CMU.EDU

### Abstract

The problem that we address is that of forecasting results by combining expert predictions. Standard ensemble methods do not explicitly consider the performance of experts to be variable across the problem domain. In contrast, we propose Specialized Weighted Majority. *Specialists* for each area of the feature space are created by augmenting experts with a classifier that chooses on which samples to vote. Our method can be seen as trading off complexity of the obtained solutions and the amount of training data required. We compare the proposed method to Weighted Majority and SVMs on synthetic and real data.

## 1. Introduction

The problem that we to address is that of expert or classifier combination, especially for applications in which the experts are humans providing forecasts. For this reason, in the following discussion we will use the word “experts” indistinctly when referring to both experts and classifiers.

We are interested in deriving a method that provides constructive interference between the experts in order to do a better job at predicting than any single expert, but also takes advantage of the characteristics of our problem domain to solve a simpler problem. One can think of learning methods as a solution to the trade-off between complexity of solution and speed of learning. On one side of the spectrum we have simple, fast learning, and fast adapting algorithms such as Winnow and

Weighted Majority (WM) voting variants which compete against the best of the component experts. On the other end we have more complex but more powerful algorithms such as SVM or AdaBoost, which compete against combinations of classifiers. This added complexity often comes at the cost of increasing the amount of training examples required to learn this complex solution.

We propose an in-between solution. Specialized Weighted Majority is a method that leverages Weighted Majority-like algorithms by modeling the performance of experts across the dimensions/features of the problem. It is known that when Weighted Majority<sup>1</sup> combines specialists (experts that are aware of their capabilities and can decide whether to vote or not), it competes against the best set of specialists rather than the best expert. In this paper we propose turning experts into specialists by directly learning their individual *areas of expertise*. Intuitively, the existence of these areas is a reasonable assumption in many cases, for example, when combining human experts.

Our goal is to retain some of the advantages of simple learning algorithms such as Weighted Majority, but at the same time being capable of finding more complex solutions. By directly learning the specialties, we hope to develop a method that trades off between both extremes of the learning spectrum.

## 2. Previous Work

Previous work on both types of learning methods (simple vs complex) is extensive. Simple methods have been shown to have good behavior against irrelevant features or noise and to be able to adapt in a timely manner to dynamical problems. Weighted Majority

---

Version of the paper for 16831-Statistical Techniques in Robotics.

<sup>1</sup>The Winnow weight-update rule is actually used in this case.

Voting (Littlestone & Warmuth, 1994) and Winnow (Littlestone, 1988; Littlestone, 1991) are the most representative algorithms in that category. Both are on-line multiplicative weight updating methods that try to minimize the total number of mistakes within the learning process. The weight adjustment process in both methods is designed so that they compete in performance with the best expert.

On the other side of the spectrum, we have ensemble methods such as AdaBoost (Freund, 1997; Schapire et al., 1998) and general purpose learners such as SVM (Burges, 1998). In a more offline fashion, both methods combine experts by optimizing the weights of each expert rather than in an incremental way with every new sample, in a more optimal and global sense with respect to all previously seen samples. By doing so, they are able to compete against linear combinations of experts.

In this study, we aim to do better than any of the experts. At the same time, we do not want to renounce to the good behavior and simplicity of Weighted Majority. With this in mind, we have devised a method that uses the concept of *specialists* to separate the learning process into two stages. In the first stage, a specialist is constructed from an expert by adding a classifier that predicts when the expert should vote or not. The second and ensembling stage uses a small modification to Weighted Majority (the Winnow weight-update rule) to combine specialists. This rule has been shown to compete against the best set of specialists (Blum, 1995; Blum, 1996), choosing the best expert for each area of the input space.

### 3. Creating specialists from experts

The problem setting that concerns us is that of emitting a prediction  $\hat{y}_t$  (from the set of classes  $C$ ) for each sample  $\mathbf{x}_t \in \mathcal{R}^D$  ( $D$  features associated to each sample). The global prediction is based on the predictions made by our  $N$  experts<sup>2</sup>,  $e^i(\mathbf{x}_t) \in C$ . Our ground truth consists of correctly labeled pairs  $(\mathbf{x}_t, y_t)$ , and we want to combine the predictions of the experts so as to minimize the number of mistakes. In standard Weighted Majority voting,  $\hat{y}_t$  is assigned such that:

$$\hat{y}_t = \arg \max_{c \in C} \sum_{i|e^i(\mathbf{x}_t)=c} w_t^i \quad (1)$$

The weights  $w_t^i$  are updated multiplicatively according to the Winnow algorithm (Littlestone, 1988; Little-

stone, 1991): if an expert makes a mistake on a sample then its weight is multiplied by a factor  $\beta \in (0..1)$ , and if (and only if) the global algorithm makes a mistake, i.e.,  $\hat{y}_t \neq y_t$ , the weights of those experts that voted correctly are divided by  $\beta$ .

The same weight-update algorithm can be applied to specialists instead of experts. In the following sections we discuss two methods to create specialists from experts. Section 3.1 describes a naive approach, while Sec. 3.2 abstracts this concept to learn a more general category of areas of expertise.

#### 3.1. Naive approach: Feature Specialists (FS)

Let us assume that the features  $\mathbf{x}_t$  describing each sample are binary. Suppose that it is also reasonable to assume that for each state of a given feature, one of the experts will perform better than all others. In this scenario, one way to construct specialists is as the Cartesian product of features and experts: for each expert and each feature, two new specialists are created; one casts a vote (the same vote as the original expert) only when the feature is active, the other only when the feature is inactive.

The set of  $N$  experts is transformed then into  $2DN$  specialists. This can be thought of as the naive approach, attempting to find the best expert for each feature independently.

#### 3.2. Specialized Weighted Majority: Explicit discovery of specialties (SWM)

In the previous method, the search for the best expert for each feature can be seen as an attempt to find under what conditions each one of the original experts has a good performance. This abstraction allows for the separation of the learning problem into two smaller sub-problems. In Specialized Weighted Majority, we first learn under what conditions the expert performs well. In a posterior stage we combine the predictions of only those experts whose learned *areas of expertise* include the specific sample under consideration. The previous decision function is modified as follows:

$$\hat{y}_t = \arg \max_{c \in C} \sum_{i|e^i(\mathbf{x}_t)=c} f^i w_t^i \quad (2)$$

Here, the filter  $f^i \in \{0, 1\}$  selects those experts that should vote, and will be a function of the features  $\mathbf{x}_t$ . In our approach, this function will be a classifier trained on previously seen examples, and will aim to predict whether the expert will be correct or not. In particular, we choose the filter to be a linear SVM decision function. See Algorithm 1 for an outline of the process.

<sup>2</sup>We can assume that the experts have much more information at their disposal than we can encode in our features  $\mathbf{x}_t$ . Making  $e^i$  a function of  $\mathbf{x}_t$  is for notational convenience only.

**Algorithm 1** Specialized Weighted Majority

---

```

1: Initialize  $w_1^i = 1$  for  $i = 1 \dots N$ 
2: for  $t = 1 \dots M$  do
3:   for  $i = 1 \dots N$  do
4:      $model_i = \text{train}(\mathbf{x}_{1:t-1}, e^i(\mathbf{x}_{1:t-1}) == \mathbf{y}_{1:t-1})$ 
5:      $f_i = \text{test}(\mathbf{x}_t, model_i)$ 
6:   end for
7:    $\hat{y}_t = \arg \max_{c \in C} \sum_{i|e^i(\mathbf{x}_t)=c} f_i w_t^i$ 
8:   Penalize classifiers with false positives: for all
    $f^i = 1$  s.t.  $e^i(\mathbf{x}_t) \neq y_t$ ,  $w_{t+1}^i \leftarrow w_t^i * \beta$ .
9:   if misclassification then
10:    Reward classifiers that predicted correctly:
    for all  $f^i = 1$  s.t.  $e^i(\mathbf{x}_t) = y_t$ ,  $w_{t+1}^i \leftarrow w_t^i * \frac{1}{\beta}$ .
11:   end if
12: end for

```

---

In the pseudo-code, the function  $train(\dots)$  represents training a classifier model on previous data with the desired output, while  $f_i = \text{test}(\mathbf{x}_t, model_i)$  evaluates this classifier on the current sample.

The naive procedure (Feature Specialists) assumes that the features cleanly partition the areas of expertise. By directly learning the expertise we are removing this assumption, and we can hope for a more accurate description of these areas.

### 3.3. Dimensionality Analysis

In posterior results, as in Figure 1, we see that Specialized Weighted Majority achieves better results than learning methods on both ends of the learning spectrum. Although the dimensionality of simple methods such as Weighted Majority is small, the performance is constrained by the best expert. On the other end, complex methods such as SVM can potentially obtain better solutions, however they are hindered by the dimensionality of the problem they try to solve.

From a dimensionality point of view, it is fair to compare SWM both with the FS approach and with SVM, representative of both categories of learning methods:

**Features Specialists** Although the algorithm remains purely as a weight multiplicative update method and the learning speed should be fast, the dimensionality of the problem jumps from  $N$  to  $2DN$ . This increases greatly the amount of data required to find a satisfying solution.

**SVM** Suppose we try to find an optimal combination of the experts by inputting to a linear SVM both the experts and the original features as features. The dimension of the problem is then  $N + D$ .

**SWM** Assuming the local classifiers for each one of the experts is implemented by a linear SVM, the number of parameters to learn is  $(D + 1)N$ . However, the separation of the problem into two stages reduces the complexity of the learning problem by decomposing it into  $N$  smaller sub-problems of dimension  $D$ .

The decomposition of the problem succeeds in creating smaller, easier sub-problems of lower dimensionality that can hopefully be solved in a quicker fashion.

## 4. Experimental Results

### 4.1. Synthetic Data

In order to evaluate and compare several methods in an ideal scenario, we have created synthetic data where the areas of expertise of each one of the experts can be specified.

#### 4.1.1. DATA GENERATION

The dataset has the following parameters:

- $N$  - The number of experts
- $K$  - The number of output classes to predict
- $M$  - The number of trials/samples.
- $D$  - The number of features associated with each sample (dimension of the input space). For this data, we have chosen binary features.

The value of the features and the output classes are generated in a purely random fashion, enforcing the absence of correlation between features  $\mathbf{x}_t$  and output  $y_t$ . The predictions of the experts for each trial are randomly generated, allowing for a certain probability of being correct for each expert. This probability depends on two parameters:  $\epsilon_g$  (capturing the *general* knowledge of the expert) and  $\epsilon_s$  (capturing the *specific expertise*, depending on the area of the input space  $\mathbf{x}_t$ ).

For any given expert, the value of its general knowledge  $\epsilon_g$  is drawn uniformly from the interval  $[0, \hat{\epsilon}_g]$ . Its specific knowledge along each one of the features of the input space is drawn uniformly from the interval  $[0, \hat{\epsilon}_s]$ . Then, for any given sample  $(\mathbf{x}, y)$ , the probability of giving a correct prediction conditioned on feature  $i$  is defined as:

$$\begin{aligned}
 P[\text{correct}|x^i = 1] &= \overbrace{\frac{1}{K}}^{P_g} + \epsilon_g + \epsilon_s \\
 P[\text{correct}|x^i = 0] &= P_g - \epsilon_s
 \end{aligned}$$

We further use conditional independence between features in order to calculate the probability of being correct on each sample as:

$$\begin{aligned}
 P[\text{correct}|\mathbf{x}] &= \frac{P[\mathbf{x}|\text{corr}]P[\text{corr}]}{P[\mathbf{x}]} \\
 &= \frac{P[\text{corr}]}{P[\mathbf{x}]} \prod_i^D P[x^i|\text{corr}] \\
 &= \frac{1}{P[\mathbf{x}]P[\text{corr}]^{D-1}} \prod_i P[x^i|\text{corr}]P[\text{corr}] \\
 &= \frac{1}{P[\mathbf{x}]P[\text{corr}]^{D-1}} \prod_i P[\text{corr}|x^i]P[x^i] \\
 P[\text{correct}|\mathbf{x}] &= \frac{1}{2^D P[\mathbf{x}]P[\text{corr}]^{D-1}} \prod_i P[\text{corr}|x^i]
 \end{aligned}$$

With a similar derivation, the overall probability of being correct (not conditioned) can be shown to be independent of  $\epsilon_s$  and equal to:

$$P[\text{correct}] = P_g = \frac{1}{K} + \epsilon_g$$

#### 4.1.2. RESULTS

In this section we compare performance with four methods: Best Expert, Weighted Majority, Feature Specialists and SVMs. For the SVM classifier, we use the LIBSVM (Chang & Lin, 2001) implementation with an RBF kernel, and all of the hyper-parameters (ie. kernel width, C) were tuned using cross-validation. In this case, both sample features  $\mathbf{x}_{1:t-1}$  and expert predictions  $e_{1:N}(\mathbf{x}_{1:t-1})$  are used as training data.

Figure 1 shows the performance measured as percentage of accumulated mistakes for a synthetic dataset with the following parameters:  $N = 20$  (experts),  $M = 200$  (samples),  $D = 10$  (features), and  $K = 2$  (classes). The y-axis varies the global knowledge of each expert ( $\epsilon_g$ ), while the x-axis shows variations in the amount of specialization of each expert  $\epsilon_s$ .

As expected, we can see that Weighted Majority is unable to learn expert specialties and its amount of mistakes is roughly aligned with the x-axis. More complicated methods (second row) are capable of taking advantage of the specialties, as can be seen in the decreasing number of mistakes across the x-axis. The complexity of SVMs requires larger values of  $\epsilon_s$  to perform as well as the proposed methods (or alternatively, a larger number of training samples). Likewise, the Feature Specialists are not well suited for this data due to the large number of active features in each sample. As hoped, our method is able to capture expert specialties and outperform the best expert.

## 4.2. Application: Sports Betting

An example problem on which to apply the proposed method is that of sports betting. The result to be predicted is the winner of a match, and the experts will be a group of sports fans offering their prediction. The main assumption is that certain people might be better at predicting the outcome of certain games, by virtue of tracking the performance of the particular teams more closely. Additionally, fans of specific teams might have unrealistic expectations or biases when their preferred team is playing.

The parameters of this dataset are:  $N = 20$  (sport fans),  $M = 273$  (matches), and  $D = 20$  (teams), and  $K = 3$  (team A wins, team B wins, or tie). Each feature  $i$  will be  $x_t^i \in \{0, 1\}$ , where 1 indicates the team  $i$  is playing and 0 otherwise. Therefore, any given sample contains 2 features set to one and 18 features set to zero.

Experiments on this dataset have shown that all algorithms perform similarly to the best expert after the 273 matches ( $142 \pm 3$  mistakes). Although, far from the worst expert (170 mistakes), the differences between algorithms are not significant enough to draw strong conclusions.

In order to understand these results, we produced a similar synthetic dataset with the same sparsity pattern (see results in Figure 2). There is a considerable decrease in the performance of SWM and SVMs, while there is an increase in the learning of FS. This can be interpreted as SVMs having more difficulty in learning from highly sparse data (due to there being less information in each sample). This directly affects the performance of our method since SVMs are used to represent the specialties. On the other hand, the simplicity of the Feature Specialists (combined with the fact that very few features are active on each sample) allows better learning.

Because the performance on the real data does not reflect the differences observed on sparse synthetic data, we believe this dataset presents a lack of clear specialties (i.e. users predict at random more often than not).

## 5. Conclusions

We have shown how turning a set of experts into specialists can lead to a significant gain. This was accomplished by augmenting experts with knowledge about their performance in different areas of the feature space. Compared to simpler combination methods such as Weighted Majority, our method offers the

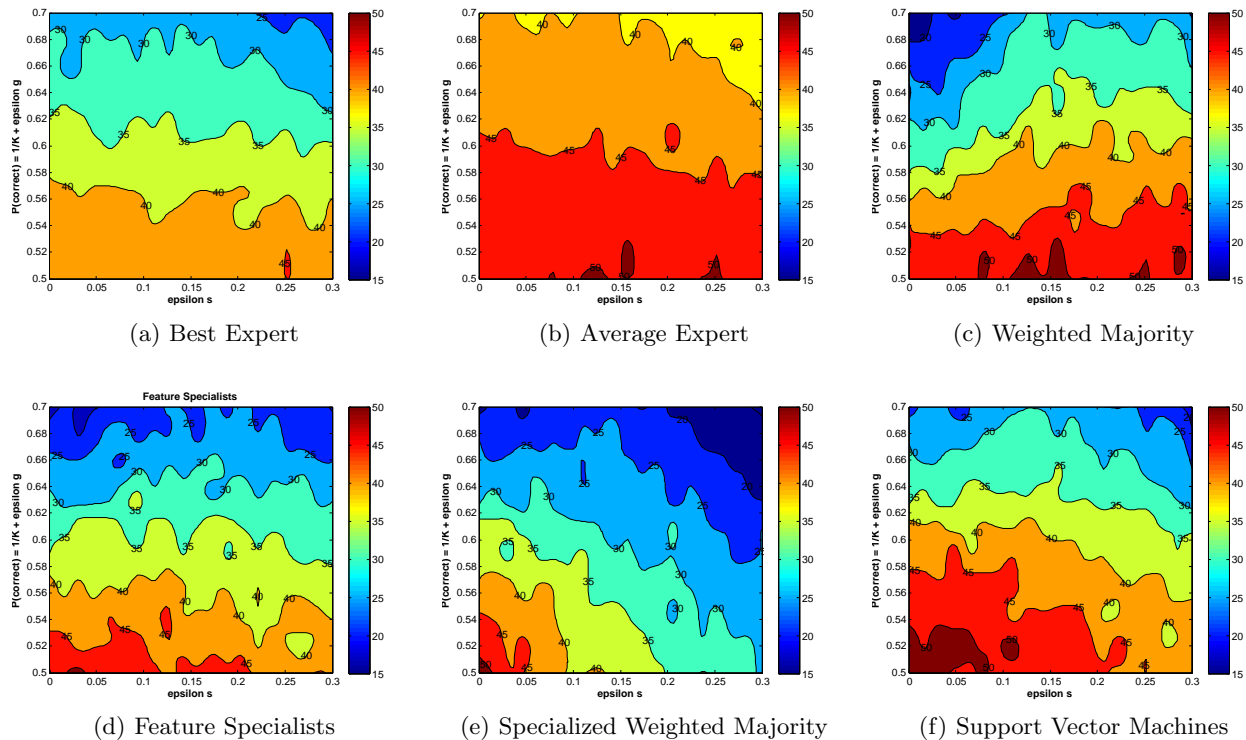


Figure 1. The figures show the percentage of accumulated mistakes.

advantage of being able to express a larger set of target concepts. Compared to more general learning methods commonly used in stacking or cascading, our method simplicity and use of domain knowledge results in faster learning (in the sense of requiring less training examples).

These conclusions have been extracted from plausibly generated (albeit still contrived) artificial data. Empirical support for our conclusions was not found in the real-data application on which we tested. Due to the small amount of available data and the similar performance of all tested methods on this set (and indeed, the similar performance of all available experts), we cannot draw strong conclusions from this experiment. It might be the case that our initial hypothesis for this data was wrong—that there are no areas of expertise in sports betting users, but it is just as plausible that not enough training data was available to learn these differences for any of the considered methods.

There are two lines of future work:

- Exploring techniques to minimize the total number of mistakes in a more global fashion. Although this would increase the dimensionality of the problem, methodologies for combining global results

with local ones have proven useful in the past. Our formulation shows a hierarchy that would easily allow such combination.

- Testing the proposed method in different applications, with more complete and suitable datasets. We have considered for other applications are weather and stock market prediction.

## References

- Blum, A. (1995). Empirical Support for Winnow and Weighted-Majority Algorithms: Results on a Calendar Scheduling Domain. *Machine Learning* (pp. 64–72). Morgan Kaufmann.
- Blum, A. (1996). On-line algorithms in machine learning. In *Proceedings of the Workshop on On-Line Algorithms, Dagstuhl* (pp. 306–325). Springer.
- Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2, 121–167.
- Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

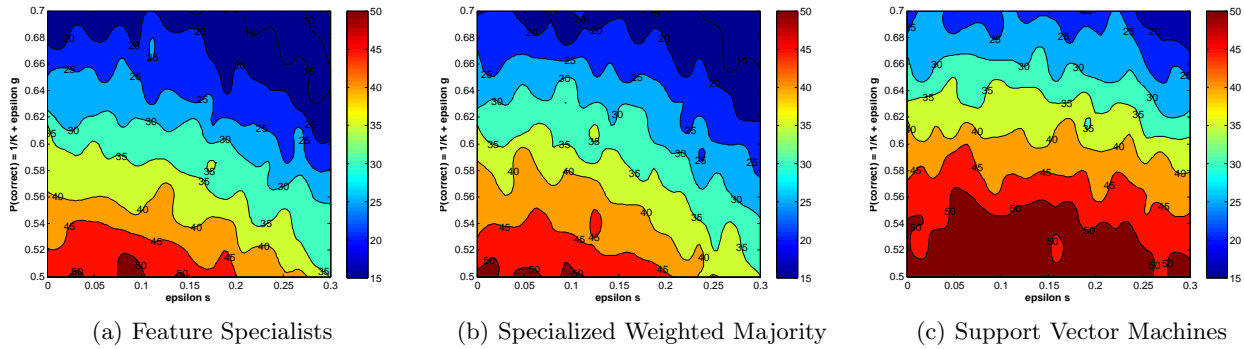


Figure 2. The above figures compare performance of the algorithms on sparse synthetic feature data, where only two features are set to 1 on any given sample.

Freund, Y. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,. *Journal of Computer and System Sciences*, 55, 119–139.

Gangardiwala, A., & Polikar, R. (2005). Dynamically weighted majority voting for incremental learning and comparison of three boosting based approaches. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 1131–1136.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning* (pp. 285–318).

Littlestone, N. (1991). Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow. *Annual Workshop on Computational Learning Theory*.

Littlestone, N., & Warmuth, M. (1994). The weighted majority algorithm. *30th Annual Symposium on Foundations of Computer Science*, 108, 256–261.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26, 1651–1686.