Boosting with Incomplete Information

Gholamreza Haffari[1*] GHAFFAR1@CS.SFU.CA
Yang Wang[1*] YWANG12@CS.SFU.CA
Shaojun Wang[2] SHAOJUN.WANG@WRIGHT.EDU
Greg Mori[1] MORI@CS.SFU.CA
Feng Jiao[3] FENGJIAO@YAHOO-INC.COM
[1]Simon Fraser University, Canada [2]Wright State University, USA [3]Yahoo! Inc. USA

Abstract

In real-world machine learning problems, it is very common that part of the input feature vector is incomplete: either not available, missing, or corrupted. In this paper, we present a boosting approach that integrates features with incomplete information and those with complete information to form a strong classifier. By introducing hidden variables to model missing information, we form loss functions that combine fully labeled data with partially labeled data to effectively learn normalized and unnormalized models. The primal problems of the proposed optimization problems with these loss functions are provided to show their close relationship and the motivations behind them. We use auxiliary functions to bound the change of the loss functions and derive explicit parameter update rules for the learning algorithms. We demonstrate encouraging results on two real-world problems — visual object recognition in computer vision and named entity recognition in natural language processing — to show the effectiveness of the proposed boosting approach.

1. Introduction

Boosting is a general supervised learning technique for incrementally building linear combinations of "weak" models to generate a "strong" predicative model. It is one of the most successful and practical methods in machine learning. Over the last decade, it has attracted much attention in the machine learning community and related areas such as statistics. It has been widely applied in many real-world problems such as text filtering and routing, ranking, learning in natural language processing, image retrieval, medical diagnosis, and customer monitoring and segmentation (Schapire, 2004).

It is very common in real-world machine learning problems that part of the input feature vector is incomplete: either not available, missing, or corrupted. In a web-page ranking problem, for example, using click-though data as part of the features, we find that a small number of valid pages have click features and most do not. In the case of object recognition in computer vision, many approaches assume a part-based model. However, certain parts of the object are hard to detect reliably due to small support in the image, occlusion or clutter, which also lead to missing information. Handling these kinds of classification problems containing incomplete information is a very important and realistic task. Excluding popular EM algorithms for generative models, some methods have been recently proposed for discriminative models (Chechik et al., 2007; Koo & Collins, 2005; Quattoni et al., 2005; Shivaswamy et al., 2006; Bi & Zhang, 2004).

In this paper, we show how to handle incomplete data under the boosting approach. We first describe the precise problem we are trying to solve, then we formulate optimization problems where the loss functions consist of two parts, one using partially labeled data and the other using fully labeled data. The primal problems of the proposed optimization problems with these loss functions are provided to show their close relationship and shed light on the rationale behind them. We derive explicit parameter update rules of the learning algorithms by introducing auxiliary functions to bound the change of loss functions. Finally, we demonstrate encouraging results on two real-world problems to show the effectiveness of the proposed boosting approach: visual object recognition in computer vision and named entity recognition in natural language processing.

[*]These authors contributed equally to this work.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

2. Preliminaries

Let $X \in \mathcal{X}$ be a random variable over data instances to be labeled, and Y be a random variable over corresponding labels ranging over a finite label alphabet $\mathcal{Y}$. The classification task is to learn a mapping from data instances $\mathcal{X}$ to labels $\mathcal{Y}$. Assume we have a set of feature functions $\mathcal{F}_1 := \{f_k(x, y)\}$ where each feature maps $\mathcal{X} \times \mathcal{Y}$ to $\mathbb{R}$. Same as in (Collins et al., 2002; Lebanon & Lafferty, 2002) and without loss of generality, we assume that the range of all feature functions in this paper is $[0, 1]$. These feature functions correspond to weak learners in boosting and sufficient statistics in an exponential family model.

Suppose the target predictor can be derived from a scoring function written as a linear combination of feature functions $t(x, y) = \sum_{f_k \in \mathcal{F}_1} \lambda_k f_k(x, y)$. Given a training dataset $\{(x_i, y_i)\}$, it has been shown (Lebanon & Lafferty, 2002) that Adaboost (Freund & Schapire, 1997) combines features to minimize the following exponential loss

$$\sum_{x_i} \sum_{y} q_\lambda(y|x_i) \tag{1}$$

where $q_\lambda(y|x) := \exp \sum_{f_k \in \mathcal{F}_1} \lambda_k \left[f_k(x, y) - f_k(x, \tilde{y}_x)\right]$ is called the unnormalized model, and $\tilde{y}_x$ denotes the label of instance x over the empirical data. Equivalently, it has been shown (Lebanon & Lafferty, 2002) that Logitboost (Friedman et al., 2000) minimizes the following log loss

$$-\sum_{x_i} \log p_\lambda(\tilde{y}_{x_i}|x_i) \tag{2}$$

where $p_\lambda(y|x) := q_\lambda(y|x)/Z_\lambda(x)$ is called the normalized model. Optimizing the two objective functions above can be done by either parallel or sequential updates (Collins et al., 2002; Lebanon & Lafferty, 2002).

Now assume that there is a random variable $h \in \mathcal{H}$ which is *hidden* in some part of the training data $\mathcal{D}_1 := \{(x_i, y_i)\}$ but has been observed in the rest of the training data $\mathcal{D}_2 := \{(x_j, h_j, y_j)\}$. Consider a second set of feature functions $\mathcal{F}_2 := \{f_k(x, h, y)\}$ where each feature maps $\mathcal{X} \times \mathcal{H} \times \mathcal{Y}$ to $\mathbb{R}$. In many real-world applications, the number of fully observed instances is much smaller than that of partially observed instances, that is, $|\mathcal{D}_2| \ll |\mathcal{D}_1|$, since obtaining fully observed instances is either expensive or time-consuming. To take full advantage of all available training data, we need to develop new methods, because the information cannot be fully exploited by the original boosting algorithm.

Hereafter we use subscripts i and j to range over training data in $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. For a datum (x, h, y), we denote all of its $\mathcal{F}_1$ features by the vector $\mathbf{f}_1(x, y)$ and all of its $\mathcal{F}_2$ features by the vector $\mathbf{f}_2(x, h, y)$.

3. Boosting with Hidden Variables

The challenge in this paper is, besides using the feature set $\mathcal{F}_1$ and training set $\mathcal{D}_1$, how to use the additional feature set $\mathcal{F}_2$ and training set $\mathcal{D}_2$ to obtain a better approximation for the mapping from instances to labels.

To this end, the main object of focus is a mapping from $\mathcal{X} \times \mathcal{H}$ to $\mathcal{Y}$, which is modeled by a conditional probability distribution $p_\lambda(y|x, h)$. This distribution is called the normalized model and is defined parametrically as

$$p_\lambda(y|x, h) \propto e^{\boldsymbol{\lambda}_1^T \cdot [\mathbf{f}_1(x,y) - \mathbf{f}_1(x,\tilde{y}_x)] + \boldsymbol{\lambda}_2^T \cdot [\mathbf{f}_2(x,h,y) - \mathbf{f}_2(x,h,\tilde{y}_x)]}$$

where $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ are the model's parameter vectors corresponding to features in $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively[1]. To estimate the parameters of the distribution, we can maximize the conditional likelihood of the training data:

$$\mathcal{L}(\lambda) := \sum_i \log p_\lambda(y_i|x_i) + \gamma \sum_j \log p_\lambda(y_j|x_j, h_j)$$

where γ is used to balance the influence of the two data sources on the objective function. Let $q_0(h|x)$ be a fixed distribution representing the prior belief in values of the hidden variable given an instance x, then $p_\lambda(y, h|x) = q_0(h|x)p_\lambda(y|x, h)$ and the first term in $\mathcal{L}(\lambda)$ can be computed based on $p_\lambda(y|x) = \sum_h p_\lambda(y, h|x)$.

We now turn our attention to model the mapping from $\mathcal{X} \times \mathcal{H}$ to $\mathcal{Y}$ by a linear scoring function that is the basis of our Adaboost type algorithms. When h is observed, the mapping is defined based on

$$t_\lambda(x, h, y) := \boldsymbol{\lambda}_1^T \cdot \boldsymbol{f}_1(x, y) + \boldsymbol{\lambda}_2^T \cdot \boldsymbol{f}_2(x, h, y)$$

and when h is hidden, it is defined as $t_\lambda(x, y) := \sum_h q_0(h|x) t_\lambda(x, h, y)$. As before, $q_0(h|x)$ is used to inject prior domain knowledge. To learn the parameters, we pose the minimization of the loss function $\mathcal{E}(\lambda)$ defined as

$$\mathcal{E}(\lambda) := \sum_i \sum_h q_0(h|x) \sum_y q_\lambda(y|x_i, h) + \gamma \sum_j \sum_y q_\lambda(y|x_j, h_j)$$

where $q_\lambda(y|x_i, h)$ is called the unnormalized model

$$q_\lambda(y|x, h) := e^{\boldsymbol{\lambda}_1^T \cdot [\mathbf{f}_1(x,y) - \mathbf{f}_1(x,\tilde{y}_x)] + \boldsymbol{\lambda}_2^T \cdot [\mathbf{f}_2(x,h,y) - \mathbf{f}_2(x,h,\tilde{y}_x)]}$$

The second term in $\mathcal{E}(\lambda)$ can be thought of as the loss incurred for the jth instance over all possible labels, and the first term as the *expected* loss for the ith instance. Note that if $q_0(h|x_j)$ puts a point mass γ on the observed h_j for instances in $\mathcal{D}_2$, then $\mathcal{E}(\lambda)$ can be rewritten compactly as

$$\mathcal{E}(\lambda) = \sum_{x \in \mathcal{D}_1 \cup \mathcal{D}_2} \sum_{h,y} q_0(h|x) q_\lambda(y|x, h)$$

[1] It is equivalent to the more familiar form $p_\lambda(x, h, y) \propto e^{\boldsymbol{\lambda}_1^T \cdot \mathbf{f}_1(x,y) + \boldsymbol{\lambda}_2^T \cdot \mathbf{f}_2(x,h,y)}$ by simply removing the constants $e^{\boldsymbol{\lambda}_1^T \cdot \mathbf{f}_1(x,\tilde{y}_x) + \boldsymbol{\lambda}_2^T \cdot \mathbf{f}_2(x,h,\tilde{y}_x)}$ from the numerator and denominator.

In the next section, we will show that there is a close relationship between minimizing $\mathcal{E}(\lambda)$ and maximizing the lowerbound $\ell(\lambda)$ on $\mathcal{L}(\lambda)$, which is derived based on Jensen's inequality and defined as

$$\ell(\lambda) := \sum_{i,h} q_0(h|x_i) \log p_\lambda(y_i|x_i, h) + \gamma \sum_j \log p_\lambda(y_j|x_j, h_j)$$

By extending q_0 to instances in $\mathcal{D}_2$ as before, we can write

$$\ell(\lambda) = \sum_{x \in \mathcal{D}_1 \cup \mathcal{D}_2} \sum_h q_0(h|x) \log p_\lambda(\tilde{y}_x|x, h)$$

Furthermore, we will show a close relationship between maximizing $\mathcal{L}(\lambda)$ and minimizing the following lowerbound on $\mathcal{E}(\lambda)$ derived by Jensen's inequality

$$\varepsilon(\lambda) := \sum_i \sum_y e^{t_\lambda(x_i,y)-t_\lambda(x_i,y_i)} \\ +\gamma \sum_j \sum_y e^{t_\lambda(x_j,h_j,y)-t_\lambda(x_j,h_j,y_j)}$$

In the test time, depending on whether h is hidden or not, either $p_\lambda(y|x)$ or $p_\lambda(y|x, h)$ can be used to determine the class label of a given instance if we use the probabilistic model. Accordingly, for the linear map, either $t_\lambda(x, y)$ or $t_\lambda(x, h, y)$ can be used.

Our definitions of both normalized and unnormalized models are similar to those in (Lebanon & Lafferty, 2002). If we ignore fully labeled data in $\mathcal{L}(\lambda)$, we get the hidden conditional random field proposed in (Koo & Collins, 2005; Quattoni et al., 2005) by assuming $q_0(h|x)$ to be constant; however, the second term in $\mathcal{L}(\lambda)$ should exist to take advantage of $\mathcal{D}_2$. If we ignore the first term in $\mathcal{E}(\lambda)$, we get the standard boosting algorithm's loss function; however, the first term is needed to take advantage of the partially observed data $\mathcal{D}_1$. In the next section, we will provide the primal problems for the proposed loss functions to motivate the rationale of optimizing them and show their relationships. We then give sequential and parallel algorithms to optimize $\mathcal{E}(\lambda)$ and $\mathcal{L}(\lambda)$ in section 5.

4. Primal and Dual Programs

It is well known (Lebanon & Lafferty, 2002) that for standard boosting with no hidden information, the primal optimization problems for Adaboost and Logitboost are the same except for the additional constraints for the latter to ensure a probabilistic model. For our boosting with incomplete information, this relationship does not exist for the original optimization problems themselves, but rather between $\mathcal{E}(\lambda)$ and $\ell(\lambda)$ which is the lowerbound on $\mathcal{L}(\lambda)$.

Let the set of non-negative measures $\mathcal{M} := \{m : \mathcal{X} \times \mathcal{H} \times \mathcal{Y} \to \mathbb{R}_+\}$, and $\mathcal{F} := \mathcal{F}_1 \cup \mathcal{F}_2$. Let $\mathbf{r}$ be the reference measure $\mathbf{1}$; however, it can be any arbitrary measure that generalizes the objective functions introduced in the previous section.

Theorem 1. *The following optimization program:*

$$\sum_{x \in D_1 \cup D_2} \sum_{h,y} q_0(h|x) q_\lambda(\tilde{y}_x|x, h) \tag{3}$$

is the dual of $\min_{\mathbf{p} \in \mathcal{S}(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F})} KL(\mathbf{p}||\mathbf{r})$ *where the extended* $KL(\mathbf{p}||\mathbf{r})$ *is defined as*

$$\sum_{x,h} \tilde{p}(x) q_0(h|x) \sum_y p(y|h, x) \Big[\log \frac{p(y|x,h)}{r(x,h,y)} - 1 \Big] + r(x, h, y)$$

and the set $\mathcal{S}(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F})$ *is defined as*

$$\Big\{ \mathbf{p} \in \mathcal{M} \Big| \sum_x \tilde{p}(x) \mathbb{E}_{q_0(h|x)p(y|x,h)} \Big[f - \mathbb{E}_{\tilde{p}(y|x)}[f] \Big] = 0, \forall f \in \mathcal{F} \Big\}$$

Proof sketch. The key idea in this theorem is the definition of the extended KL divergence and $\mathcal{S}(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F})$. Construct the Lagrangian of the dual, which is a constrained optimization problem, take its derivative, and set it to zero. It will give the form of the optimal solution; plug this form back into the Lagrangian, and make the data consistency assumption ($\tilde{\mathbf{p}}$ is the *empirical* probability distribution) $\sum_y \tilde{p}(y|x) f(x, y) = f(x, y_x)$ for $f \in \mathcal{F}_1$ and $\sum_y \tilde{p}(y|x) f(x, h, y) = f(x, h, y_x)$ for $f \in \mathcal{F}_2$, we will obtain the optimization problem in (3) . □

Theorem 2. *The following optimization program:*

$$\sum_{x \in D_1 \cup D_2} \sum_h q_0(h|x) \log p_\lambda(\tilde{y}_x|x, h) \tag{4}$$

is the dual of $\min_{\mathbf{p} \in \mathcal{S}_\Delta(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F})} KL(\mathbf{p}||\mathbf{r})$ *where the extended* $KL(\mathbf{p}||\mathbf{r})$ *is defined as in Theorem 1, and*

$$\mathcal{S}_\Delta(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F}) := \Big\{ \mathbf{p} \in \mathcal{S}(\tilde{\mathbf{p}}, \mathbf{q}_0, \mathcal{F}) \Big| \forall x, h : \sum_y p(y|x, h) = 1 \Big\}$$

The proof of this theorem is similar to that of Theorem 1 and is omitted because of space constraints. As can be seen from the theorems above, the primal optimization problems corresponding to the objective functions $\mathcal{E}(\lambda)$ and $\ell(\lambda)$ are the same except for the additional constraints for the later one to ensure $\sum_y p(y|x, h) = 1$. The extended KL divergence gives the expected discrepancy between $p(y|x, h)$ and the reference measure $r(x, h, y)$ where the expectation is taken with respect to the distribution $\tilde{p}(x) q_0(h|x)$. Hence minimizing the extended KL subject to the constraints forces $p(y|x, h)$ to become similar to $\mathbf{r}$, or in particular when the reference measure is $\mathbf{1}$ or constant, to have more entropy.

5. Learning Algorithms

Convergence of boosting algorithms has been studied in various ways. Much work has been done to prove the convergence in terms of an optimization method, which can be categorized into two approaches: greedy function optimization and greedy feature induction.

In the first approach, the boosting algorithm is viewed as a sequential gradient descent algorithm (Breiman, 1999;

Algorithm 1 Parallel Updates for the Normalized Model

```
repeat
  for $f_k \in \mathcal{F}_1$ do
    $A_k^+ = \sum_i \mathbb{E}^+_{p_\lambda(y|x_i)}[g_k(x_i,y)] + \gamma \sum_j \mathbb{E}^+_{p_\lambda(y|x_j,h_j)}[g_k(x_j,y)]$
    $A_k^- = \sum_i \mathbb{E}^+_{p_\lambda(y|x_i)}[-g_k(x_i,y)] + \gamma \sum_j \mathbb{E}^+_{p_\lambda(y|x_j,h_j)}[-g_k(x_j,y)]$
    $\Delta\lambda_k = \frac{\log A_k^- - \log A_k^+}{2C}$
  end for
  for $f_k \in \mathcal{F}_2$ do
    $A_k^+ = \sum_i \mathbb{E}^+_{p_\lambda(y,h|x_i)}[g_k(x_i,h,y)] + \gamma \sum_j \mathbb{E}^+_{p_\lambda(y|x_j,h_j)}[g_k(x_j,h_j,y)]$
    $A_k^- = \sum_i \mathbb{E}^+_{p_\lambda(y,h|x_i)}[-g_k(x_i,h,y)] + \gamma \sum_i \mathbb{E}^+_{p_\lambda(y|x_j,h_j)}[-g_k(x_j,h_j,y)]$
    $\Delta\lambda_k = \frac{\log A_k^- - \log A_k^+}{2C}$
  end for
  for $f_k \in \mathcal{F}_1 \cup \mathcal{F}_2$ do
    $\lambda_k \leftarrow \Delta\lambda_k + \lambda_k$
  end for
until convergence
```

Friedman et al., 2000; Mason et al., 2000) in function space, inspired by numerical optimization and statistical estimation. It is a forward stage-wise additive modeling that approximates the solution by sequentially adding new basis functions without adjusting the parameters and coefficients of those that have already been added. At each iteration, one solves for the optimal basis function and corresponding coefficients to add to the current expansion. This produces new expansion, and the process is repeated.

In the second approach (Collins et al., 2002; Lebanon & Lafferty, 2002), the boosting algorithm is described as a greedy feature induction algorithm to incrementally build random fields. The greediness of the algorithm arises in steps that select the most informative feature. In these steps each feature in a pool of candidate features is evaluated by estimating the reduction in the Kullback-Lieber divergence that would result from adding the feature to the field. This reduction is approximated as a function of a single parameter and is equal to the exponential loss reduction or log loss increment. This approximation is one of the key elements that make it practical to evaluate a large number of candidate features at each stage of the induction algorithm. Various parameter update rules can be derived By using an auxiliary function to bound the change of loss function from above, and thus convergence to the global optimal solution is proved.

In this paper we the second approach to learn the discriminative model. We construct an auxiliary function to bound the change of exponential loss, $\mathcal{E}(\lambda + \Delta\lambda) - \mathcal{E}(\lambda)$ or logloss $\mathcal{L}(\lambda) - \mathcal{L}(\lambda + \Delta\lambda)$. Similar to (Collins et al., 2002; Lebanon & Lafferty, 2002), either parallel or sequential updates can be used. By the same argument as in (Collins et al., 2002; Lebanon & Lafferty, 2002), we can show the convergence of these updates to a *local minimum* of the loss function. For simplicity in presenting the results, we introduce some notation for $\tilde{x} \in \mathcal{D}_1 \cup \mathcal{D}_2$:

$$\forall f_k \in \mathcal{F}_1, g_k(\tilde{x}, y) = f_k(\tilde{x}, y) - f_k(\tilde{x}, \tilde{y}_{\tilde{x}}) \quad (5)$$

$$\forall f_k \in \mathcal{F}_2, g_k(\tilde{x}, h, y) = f_k(\tilde{x}, h, y) - f_k(\tilde{x}, h, \tilde{y}_{\tilde{x}}) \quad (6)$$

$$C := \max_{\tilde{x},y,h} \Big(\sum_{f_k \in \mathcal{F}_1} |g_k(\tilde{x}, y)| + \sum_{f_k \in \mathcal{F}_2} |g_k(\tilde{x}, h, y)| \Big) \quad (7)$$

$$\mathbb{E}^+_{p(t)}[\psi(t)] := \sum_{t:\psi(t)>0} p(t)\psi(t) \quad (8)$$

For the normalized model, the learning algorithm with parallel updates is summarized in Algorithm 1 and with the sequential updates in Algorithm 2. For the unnormalized model, the update rules (parallel or sequential) are exactly the same; the only difference is that we will use $q_\lambda(y|x,h)$ rather than $p_\lambda(y|x,h)$ in all the algorithms' equations. For details of the derivation of updating rules in the learning algorithms, see Appendix A.

For ease of presentation, we have assumed that the potentially missing attributes are always the same. This is an interesting and nontrivial situation that occurs in many real-world applications, where the missing attribute h is the information that requires expensive human labeling (see the experiments for example applications). However, our approach can be easily extended to the cases where the data could have different missing attributes. In this more general setting, the i-th training datum has the form (x_i, y_i) with missing information $h_i \in \mathcal{H}_i$, where $\mathcal{H}_i$ can vary for different i's depending on which information is missing. The contribution of this datum to the log loss in the normalized model is simply $-\log p_\lambda(y_i|x_i)$. All the arguments in this paper will go through with some minor changes.

6. Experiments

We evaluate our approach in two real-world problems: visual object recognition in computer vision and named entity recognition in natural language processing. In both cases, we use simple and independent features, so when we calculate the values of A_j^+ and A_j^-, feature expectations can be done efficiently. For simplicity, we set γ to be 1. In practice, this parameter can be set by cross-validation. We set our prior belief in values of the hidden variable given an

Algorithm 2 Sequential Updates for the Norm. Model

```
repeat
  for $f_k \in \mathcal{F}_1$ do
    $A_k^+ = \sum_i \sum_{y \neq y_i} p_\lambda(y|x_i)(1 + g_k(x_i, y)) + \gamma \sum_j \sum_{y \neq y_j} p_\lambda(y|x_j, h_j)(1 + g_k(x_j, y))$
    $A_k^- = \sum_i \sum_{y \neq y_i} p_\lambda(y|x_i)(1 - g_k(x_i, y)) + \gamma \sum_j \sum_{y \neq y_j} p_\lambda(y|x_j, h_j)(1 - g_k(x_j, y))$
    $\lambda_k \leftarrow \frac{\log A_k^- - \log A_k^+}{2} + \lambda_k$
  end for
  for $f_k \in \mathcal{F}_2$ do
    $A_k^+ = \sum_i \sum_{y \neq y_i} \sum_h p_\lambda(y, h|x_i)(1 + g_k(x_i, h, y)) + \gamma \sum_j \sum_{y \neq y_j} p_\lambda(y|x_j, h_j)(1 + g_k(x_j, h_j, y))$
    $A_k^- = \sum_i \sum_{y \neq y_i} \sum_h p_\lambda(y, h|x_i)(1 - g_k(x_i, h, y)) + \gamma \sum_j \sum_{y \neq y_j} p_\lambda(y|x_j, h_j)(1 - g_k(x_j, h_j, y))$
    $\lambda_k \leftarrow \frac{\log A_k^- - \log A_k^+}{2} + \lambda_k$
  end for
until convergence
```

instance, $q_0(h|x)$ to be constant. In all the experiments, we use parallel updates. We have tried sequential updates and find that they are much slower. Although they can achieve higher likelihood on the training data, the results on the test data remain the same.

We compare our proposed boosting approach with three different baseline algorithms, in both normalized and unnormalized cases. The first baseline algorithm (BL1) uses both sets of features $\mathcal{F}_1$ and $\mathcal{F}_2$, but is trained only on the fully observed training data $\mathcal{D}_2$. The second baseline algorithm (BL2) is trained on all the training data $\mathcal{D}_1 \cup \mathcal{D}_2$ but uses only features $\mathcal{F}_1$, that is, it ignores features $\mathcal{F}_2$ that involve the hidden information h. Notice that the second baseline algorithm is identical to the algorithm in (Lebanon & Lafferty, 2002). The third baseline algorithm (BL3) uses all the training data $\mathcal{D}_1 \cup \mathcal{D}_2$ and both types of features $\mathcal{F}_1 \cup \mathcal{F}_2$ but ignores observed h on fully observed data; that is, it assumes all the data are in the form of $\{(x_i, y_i)\}$. Notice that the third baseline algorithm is similar to the hidden conditional random field (Quattoni et al., 2005).

6.1. Visual Object Recognition

We first consider a visual object recognition task where some of the data have missing features. In this task, we attempt to classify an image based on the existence of an object of interest in the image. We test our approach on the Caltech 4 dataset: airplanes, cars, faces, and motorbikes.

Common approaches to object recognition involve some form of supervision, which may range from manually segmenting the objects (Winn & Shotton, 2006), to specifying a bounding box of the objects (Viola & Jones, 2001), to only indicating the existence of the objects (Fergus et al., 2003). Naturally, there is a trade-off among different levels of supervisions. Manually segmenting the object of interest in an image obviously provides very accurate information for any learning algorithm, but it is very expensive and time-consuming to annotate a large number of images. On the other hand, it is relatively easy to label an image based only on the existence of an object. In our experiment, we assume we have two sets of training images. The first set of images has only class labels associated with them; we represent them as (x, y), where x refers to the image and y refers to its class label. The second set of images has both class label and the contour of the object being manually labeled; we represent them as (x, h, y), where h is the information about the contour of the object. Our learning problem is then in precisely the scenario in which our proposed method is expected to be effective.

We first run an interest-point detector (Kadir & Brady, 2001) to identity regions of interest on each image. Each interest point is represented by a SIFT descriptor (Lowe, 2004) as a 128-dimensional vector. The SIFT descriptors from all the training images are then vector quantized into K visual words (we choose $K = 200$ in our experiment) by k-means clustering. All the images are then represented by a bag-of-words representation by counting the occurrence of each visual word in an image. We denote an image as $x = (x_1, x_2, ..., x_t)$, where t is the number of interest points in x, and each x_i is an entry to a visual word. The information h about the object contour is represented as $h = (h_1, h_2, ..., h_t)$, where h_i is a binary value indicating whether x_i is on the object or not. Since we assume the "bag-of-words" model, the summation over h required for calculating A_j^+ and A_j^- can be solved efficiently by factoring out the contribution of each interest point. Although bag-of-words representation ignores a lot of positional information between features, previous work (Sivic et al., 2005; Fergus et al., 2005) has demonstrated that it to be quite effective in object recognition tasks.

We define the following three sets of features for our boosting algorithm, based on the bag-of-words representation of images. (1) feature $f_{jy'}(x, y)$ is calculated as the count of visual words j in an image x if $y = y'$, and zero otherwise; (2) feature $o_{jy'}(x, h, y)$ is the count of visual words j on the foreground of image x if $y = y'$, and zero otherwise; (3) feature $b_{jy'}(x, h, y)$ is the count of visual words j on the background of image x if $y = y'$, and zero otherwise. Notice that features $f_{jy'}$ are always observed for a training image. Features $o_{jy'}$ and $b_{jy'}$ are observed only when a training image does not have missing information (i.e., the manually labeled object contour). We normalize all the features by the total number of interest points in an image

	accuracy	log-likelihood
Our method	97.22%	-0.0916
BL1	89.26%	-1.1417
BL2	88.01%	-0.5698
BL3	90.43%	-0.4375

normalized model

	accuracy	log of loss
Our method	94.83%	-0.7412
BL1	82.57%	-1.1231
BL2	89.86%	-0.7977
BL3	87.64%	-0.8068

unnormalized model

Table 1. Results of our approach on visual object recognition, compared with three baseline algorithms

to make sure their values are between 0 and 1. During testing, we observed the image x, and we try to infer its label y based on the learned model. Although we can also infer y assuming both x and h are observed during testing, it is actually an unrealistic setting in our application. It requires a perfect figure/ground segmentation of the image x. However, since figure/ground segmentation is itself a very challenging problem in computer vision, it is not reasonable to assume we could have this information during the testing. So we do not investigate this case.

Our dataset contains more than 2000 images. We randomly split them equally into training and testing sets. We choose 30% of the training images to be fully observed and the rest to be partially observed. We compare both normalized and unnormalized models with the three baseline algorithms defined above, in terms of classification accuracy and the log-likelihood of the test data. The results are shown in Table 1. We find that our approach is significantly superior to the three baseline algorithms, in term of both accuracy and log-likelihood on the test images.

6.2. Named Entity Recognition

Named entity extraction (NEE) is a subtask of information extraction in which we try to identify names of persons, locations, and organizations in a given set of documents. One approach to this problem is to do first named entity recognition (NER) and then named entity classification (NEC). In this section we apply our method to the NER problem and demonstrate its effectiveness compared to the baseline systems.

We consider NER as a sequence labeling problem, that is, specifying a sequence of zero and one for a sentence to classify a word as part of a named entity or not. For each word w, its surrounding words in a window of length 5, its part-of-speech tag (when available), and previous predictions represent its local context, which then used by the classifier. The part-of-speech tag is a valuable source of information and is not available in some annotations of the data sets for this task, so we treat it as the hidden variable that is not observed for some portion of the training data. We could use the *sequence* of POS tags of the words in the current window as the hidden variable. In that case, we may use a finite state automata to characterize the eligible sequence of POS tags when we want to sum over their values to speed up the training algorithms. The features that we used are summarized in Table 6.2; they are described in more details in (Carreras et al., 2003).

Feature	Explanation
Lexical	word forms and their positions in the window
Syntactic	part-of-speech tags (when available)
Orthographic	capitalized, include digits, ...
Affixes	the suffixes and prefixes (up to four characters)
Left predict	predicted labels for the two previous words

Table 2. Details of the features used for the NER task. Syntactic features belong to $\mathcal{F}_2$ and the rest of features belong to $\mathcal{F}_1$.

We use the data set of the CONLL 2003 shared task. To reduce the training time, we collapse the original 45 different POS tags into five tags as done in (McCallum et al., 2003). After training the model, we do the classification for each individual position by normalizing the prediction score of the model using the class mass normalization (CMN) procedure as introduced in (Zhu et al., 2003).

We compare our approach to the three baseline systems defined before. There are 5K sentences in $\mathcal{D}_1$, 6K sentences in D_2, and 1K sentences in the test set. The first set of experiments show the performance of our model compared to the baselines when, at the test time, only x is available (see Table 3). In the second set of experiments, (x, h) is given at the test time (see Table 4); for this setting, BL2 and BL3 cannot be used. Our method outperforms baseline systems in both sets of experiments in terms of f-measure and log-likelihood or loss function.

7. Comparison to the Related Work

Originally boosting is considered as a way to boost weak learners to strong learners by: learning weak hypotheses to classify hard examples in each round, and finally combining these weak hypotheses. Another view to boosting is through the statistical perspective which interprets it as: optimizing some objective function via parallel or sequential updates to determine the weights of all possible weak hypotheses (aka features). There is a debate between the statistic and algorithmic perspective; see (Mease & Wyner, 2008) for more information. Our work takes the statistical perspective and do not engage in that debate.

A related algorithm that takes the first perspective to boost-

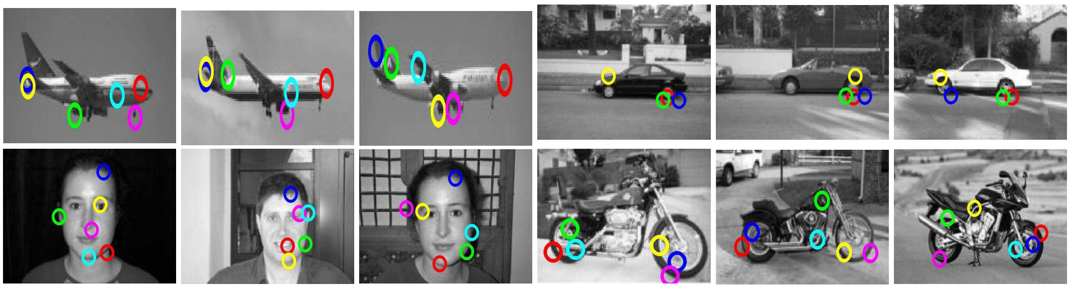

Figure 1. Visualization of most discriminative patches in each image.

	f-measure	log-likelihood
Our method	49.45%	-0.5784
BL1	46.63%	-0.5932
BL2	48.10%	-0.5803
BL3	47.80%	-0.5880

normalized model

	f-measure	log of loss
Our method	49.04%	-2.6337
BL1	46.24%	-2.6458
BL2	47.58%	-2.6378
BL3	46.39%	-2.6434

unnormalized model

Table 3. Results of our approach on the NER task, compared with three baseline algorithms when only x is given in the test data.

	f-measure	log-likelihood
Our method	59.60%	-0.5759
BL1	56.51%	-0.5916

normalized model

	f-measure	log of loss
Our method	60.17%	-0.2586
BL1	55.46%	-0.2655

unnormalized model

Table 4. Results of our approach on the NER task, compared with the baseline algorithm BL1 when (x, h) is given in the test data. Even by having h, namely POS tags, the NER task is not easy.

ing is AdaBoost with confidence-rated predictions in which a weak learner outputs a real value representing the confidence level (Schapire & Singer, 1999). When provided with incomplete input, the weak learner's contribution is its uncertainty about its vote, which is represented by the produced real number. The details of the connection between our approach and confidence-rated AdaBoost will be an interesting topic to explore for future research.

8. Conclusions and Further Work

In this work we have presented a novel boosting approach that extends the traditional boosting framework by incorporating hidden variables such that fully labeled data can be integrated with partially labeled data to form a powerful strong classifier. Thus, compared with both the original boosting algorithms and hidden CRF, our model performs better in two real-world problems by fully exploiting relevant complete information of data resources.

We consider only simple independent features in our model. In fact, the hidden variables may have complex dependencies that respect certain cyclic graph structure; then it may be necessary to use variational methods, such as loopy belief propagation, to compute feature expectation for the values of A^+ and A^-. As future work, we would like to incorporate more complex dependent features in these two applications.

Acknowledgment

SW is supported in part by the Research Challenge Grant at Wright State University. We would like to thank Anoop Sarkar, David Blei and anonymous reviewers for their constructive comments.

References

Bi, J., & Zhang, T. (2004). Support vector classification with input data uncertainty. *NIPS*.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation*.

Carreras, X., Màrquez, L., & Padró, L. (2003). A simple named entity extractor using AdaBoost. *Proceedings of CoNLL*.

Chechik, G., Heitz, G., Elidan, G., Abbeel, P., & Koller, D. (2007). Max-margin classification of incomplete data. *NIPS*.

Collins, M., Schapire, R. E., & Singer, Y. (2002). Logistic regression, adaboost and bregman distances. *Machine Learning*.

Fergus, R., Fei-Fei, L., Perona, P., & Zisserman, A. (2005). Learning object categories from google's image search. *IEEE International Conference on Computer Vision*.

Fergus, R., Perona, P., & Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. *CVPR*.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55(1):119-139*.

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, *28*, 337–407.

Kadir, T., & Brady, M. (2001). Scale, saliency and image description. *International Journal of Computer Vision*, *45*, 83–105.

Koo, T., & Collins, M. (2005). Hidden-variable models for discriminative reranking. *Proceedings of EMNLP*.

Lebanon, G., & Lafferty, J. D. (2002). Boosting and maximum likelihood for exponential models. *NIPS*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (2000). Functional gradient techniques for combining hypotheses. In *Advances in large margin classifiers*, 221–246. MIT Press.

McCallum, A., Rohanimanesh, K., & Sutton, C. (2003). Dynamic conditional random fields for jointly labeling multiple sequences. *NIPS Workshop on Syntax, Semantics, Statistics*.

Mease, D., & Wyner, A. (2008). Evidence contrary to the statistical view of boosting. *JMLR*, *9*.

Quattoni, A., Collins, M., & Darrell, T. (2005). Conditional random fields for object recognition. *NIPS*.

Schapire, R. (2004). The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*. Springer.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*, 297–336.

Shivaswamy, P. K., Bhattacharyya, C., & Smola, A. J. (2006). Second order cone programming approaches for handling missing and uncertain data. *JMLR*, *7*.

Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., & Freeman, W. T. (2005). Discovering objects and their location in images. *IEEE International Conference on Computer Vision*.

Viola, P., & Jones, M. (2001). Robust real-time object detection. *Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling*.

Winn, J., & Shotton, J. (2006). The layout consistent random field for recognizing and segmenting partially occluded objects. *CVPR*.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *ICML*.

Appendix A. Deriving Learning Algorithms

Exponential loss

We derive parallel updates for exponential loss. Let $\lambda + \Delta\lambda$ be the new parameters value. We find an upper-bound to the change of objective function $\mathcal{E}(\lambda + \Delta\lambda) - \mathcal{E}(\lambda)$ by an *auxiliary function*, and then minimize the bound.

$$\begin{aligned}
&\mathcal{E}(\lambda + \Delta\lambda) - \mathcal{E}(\lambda) = \\
&\sum_{i,h,y} q_0(h|x_i) e^{\lambda . G(x_i,h,y)} \left(e^{\Delta\lambda . G(x_i,h,y)} - 1 \right) + \\
&\gamma \sum_{j,y} e^{\lambda . G(x_j,h_j,y)} \left(e^{\Delta\lambda . G(x_j,h_j,y)} - 1 \right) \leq \\
&\sum_{i,h,y} q_\lambda(h,y|x_i) \Big(\frac{\sum_k g_k e^{\Delta\lambda s_k(x_i,h,y) C}}{C} + w_{x_i,h,y} - 1 \Big) + \\
&\gamma \sum_{j,y} q_\lambda(y|x_j,h_j) \Big(\frac{\sum_k g_k e^{\Delta\lambda s_k(x_j,h_j,y) C}}{C} + w_{x_j,h_j,y} - 1 \Big) \\
&:= \mathcal{A}(\lambda, \Delta\lambda)
\end{aligned}$$

where $w_{x,y,h} = 1 - \sum_k \frac{|g_k(x,h,y)|}{C}$, $G(x,h,y)$ is a vector built from $g_k(x,h,y)$, and $s_k(x,h,y)$ is the sign of $g_k(x,h,y)$. We find the stationary point of the auxiliary function $\mathcal{A}(\lambda, \Delta\lambda)$ with respect to $\Delta\lambda_k$ by taking the derivative and setting it to zero, which gives us the updating rules. The sequential updates can also be derived similarly.

Log loss

The objective is to minimize the objective function $-\mathcal{L}(\lambda + \Delta\lambda) + \mathcal{L}(\lambda)$. First we find an upper-bound on the objective function:

$$\begin{aligned}
&\mathcal{L}(\lambda) - \mathcal{L}(\lambda + \Delta\lambda) = \\
&\sum_i \log \sum_{y,h} e^{\lambda + \Delta\lambda \cdot G(x_i,h,y)} - \log \sum_{y,h} e^{\lambda \cdot G(x_i,h,y)} - \\
&\gamma \sum_j \log \sum_y e^{\lambda + \Delta\lambda \cdot G(x_j,h_j,y)} - \log \sum_y e^{\lambda \cdot G(x_j,h_j,y)} = \\
&\sum_i \log \sum_{y,h} p_\lambda(h,y|x) e^{\Delta\lambda \cdot G(x_i,h,y)} + \\
&\gamma \sum_j \log \sum_y p_\lambda(y|x_j,h_j) e^{\Delta\lambda \cdot G(x_j,h_j,y)} \leq \\
&\sum_i \sum_{y,h} p_\lambda(h,y|x) e^{\Delta\lambda \cdot G(x_i,h,y)} \\
&+ \gamma \sum_j \sum_y p_\lambda(y|x_j,h_j) e^{\Delta\lambda \cdot G(x_j,h_j,y)}
\end{aligned}$$

The inequality holds because $\log x \leq x$. The last expression can be upper-bounded again (using a similar technique used for exponential loss), and the resultant upper-bound will be the auxiliary function. It can be shown that the update rules are the same to unnormalized model, but the difference is to use $p_\lambda(\cdot)$ rather than $q_\lambda(\cdot)$. The update rules for sequential updates can be derived similarly.