Chapter 8

Boosting, Convex Optimization and Information Geometry

In the last chapter, we saw how AdaBoost can be viewed as a special case of more general methods for optimization of an objective function, namely coordinate descent and functional gradient descent. In both of those cases, the emphasis was on construction of a final classifier by manipulating the weights defining it, or equivalently, by iteratively adding multiples of base functions to it. In this chapter, we present a rather different perspective on the behavior and dynamics of AdaBoost. In this new view, the emphasis will instead be on the distributions D_t, that is, on the weights over examples, which exist in a dual universe to the weights on the weak classifiers that define the final hypothesis.

As before, we will see that AdaBoost is a special case of a more general and much older algorithm, namely, a method that combines both geometry and information theory in which the distributions D_t are regarded as points in $\mathbb{R}^m$ that are repeatedly projected onto hyperplanes defined by the weak hypotheses. This new approach will lend geometric intuition to the workings of AdaBoost, revealing a beautiful mathematical structure underlying the workings of the algorithm. We will see that, viewed in terms of optimization, AdaBoost is in fact solving two problems simultaneously, one being the minimization of exponential loss as discussed in Section 7.1, and the other being a dual optimization problem involving maximization of the entropy of the distributions over examples subject to constraints. This understanding will enable us to answer basic questions about AdaBoost's dynamics, particularly with regard to the convergence of the example distributions, thus providing the means to prove that the algorithm asymptotically minimizes exponential loss.

The framework we present encompasses logistic regression as well, with only

minor modification, thus providing further and deeper unification.

We end this chapter with an application to the problem of modeling the habitat of plant and animal species.

8.1 Iterative projection algorithms

In this section, we explain how AdaBoost can be seen to be a kind of geometric *iterative projection algorithm*. Although we will eventually be working using a geometry based on notions from information theory, we begin by considering the analogous setting for ordinary Euclidean geometry.

8.1.1 The Euclidean analog

To get an intuitive feeling for such algorithms, consider the following simple geometric problem. Suppose we are given a set of linear constraints on points $\mathbf{x}$ in $\mathbb{R}^m$:

$$\begin{aligned} \mathbf{a}_1 \cdot \mathbf{x} &= b_1 \\ \mathbf{a}_2 \cdot \mathbf{x} &= b_2 \\ &\vdots \\ \mathbf{a}_n \cdot \mathbf{x} &= b_n. \end{aligned} \tag{8.1}$$

These n constraints define a linear subspace of $\mathbb{R}^m$, namely

$$\mathcal{P} \doteq \{\mathbf{x} \in \mathbb{R}^m : \mathbf{a}_j \cdot \mathbf{x} = b_j \text{ for } j = 1, \ldots, n\}, \tag{8.2}$$

which we assume to be non-empty and which we refer to as the *feasible set*. We are also given a *reference point* $\mathbf{x}_0 \in \mathbb{R}^m$. The problem is to find the point $\mathbf{x}$ satisfying the n constraints that is *closest* to $\mathbf{x}_0$. Thus, the problem is to find $\mathbf{x}$ solving the following constrained optimization problem:

$$\begin{aligned} &\text{minimize } \|\mathbf{x} - \mathbf{x}_0\|_2^2 \\ &\text{subject to } \mathbf{a}_j \cdot \mathbf{x} = b_j \text{ for } j = 1, \ldots, n. \end{aligned} \tag{8.3}$$

This is the first of several *convex programs* to be presented in this chapter, where in each the goal is to minimize a convex function subject to linear constraints. Although there are surely many ways of attacking this problem, here is a simple approach that turns out to be rather general. Start at $\mathbf{x} = \mathbf{x}_0$. If all the constraints are satisfied, then we are done. Otherwise, select one unsatisfied constraint, say $\mathbf{a}_j \cdot \mathbf{x} = b_j$, and *project* $\mathbf{x}$ onto the hyperplane defined by this equality; that is, replace

Given: $\mathbf{a}_j \in \mathbb{R}^m$, $b_j \in \mathbb{R}$ for $j = 1, \ldots, n$
$\mathbf{x}_0 \in \mathbb{R}^m$
Find: $\mathbf{x} \in \mathcal{P}$ minimizing $\|\mathbf{x} - \mathbf{x}_0\|_2^2$ where $\mathcal{P}$ is as in Eq. (8.2)
$\mathbf{x}_1 = \mathbf{x}_0$
For $t = 1, 2, \ldots$ until $\mathbf{x}_t$ is "close enough" to $\mathcal{P}$

- choose a constraint j
- let $\mathbf{x}_{t+1} = \arg \min_{\mathbf{x}:\mathbf{a}_j \cdot \mathbf{x} = b_j} \|\mathbf{x} - \mathbf{x}_t\|_2^2$

Output $\mathbf{x}_t$

Figure 8.1: An iterative projection algorithm for finding the closest point, subject to linear constraints.

$\mathbf{x}$ by the *closest* point satisfying $\mathbf{a}_j \cdot \mathbf{x} = b_j$. Now repeat "until convergence." Pseudocode for this technique is given in Figure 8.1.

A simple three-dimensional example is shown in Figure 8.2. Here, there are only two constraints depicted by the two planes. The feasible set $\mathcal{P}$ is the line at their intersection. Beginning at $\mathbf{x}_0$, the algorithm repeatedly projects onto one plane and then the other, quickly converging to a point in $\mathcal{P}$ which turns out to be the point closest to $\mathbf{x}_0$.

We did not specify how to choose a constraint j on each iteration of Figure 8.1. One idea is to simply cycle through the constraints in order. Another option is to greedily choose the hyperplane $\mathbf{a}_j \cdot \mathbf{x} = b_j$ that is *farthest* from $\mathbf{x}_t$ with the expectation that bringing $\mathbf{x}_{t+1}$ into line with this most violated constraint will effect the greatest and quickest progress. We will generally focus on this latter option which we refer to as *greedy selection*.

It can be proved that with either of these options, as well as a host of others, this algorithm is guaranteed to converge to the (unique) solution, provided the feasible set $\mathcal{P}$ is non-empty. When $\mathcal{P}$ is empty, the problem is no longer meaningful, but the iterative projection algorithm can still be executed. The algorithm cannot converge to a single point since there will always remain unsatisfied constraints. Beyond this observation, the dynamics of this simple procedure are not fully understood in this case, especially when using greedy selection.

8.1.2 AdaBoost as an iterative projection algorithm

AdaBoost, it turns out, is also an iterative projection algorithm that operates in the space of probability distributions over training examples. To see this, we will need to spell out the appropriate linear constraints, but we also will need to modify how

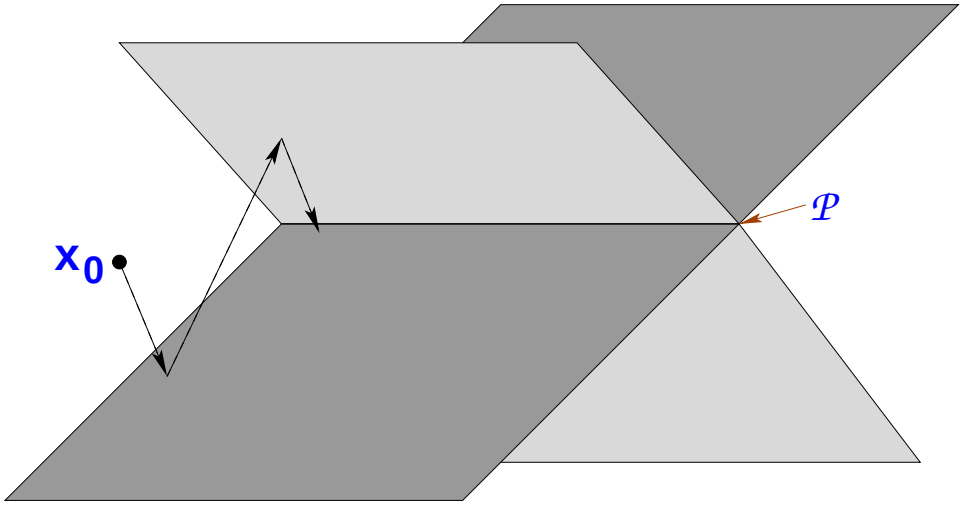

Figure 8.2: An example in $\mathbb{R}^3$ of the input to an iterative projection algorithm. Here, the problem is to find the point at the intersection of the two planes that is closest to the reference point $\mathbf{x}_0$.

we measure distance between points.

As usual, we are given a dataset $(x_1, y_1), \ldots, (x_m, y_m)$, and a space of weak classifiers

$$\mathcal{H} = \{\hbar_1, \ldots, \hbar_n\}.$$

For simplicity, we can assume that these are all binary $\{-1, +1\}$-valued, although this is quite unimportant to what follows. We follow the notation of Figure 1.1. Unlike in earlier parts of the book, our focus now is on probability distributions D over $\{1, \ldots, m\}$, the indices of the m training examples, in other words, the sort of probability distributions D_t used by AdaBoost.

What sorts of linear constraints on these distributions does AdaBoost seem to want satisfied? We saw in Section 6.4 that in a game-theoretic context in which the boosting algorithm opposes the weak learning algorithm, the booster's goal is to find a distribution over examples that is as hard as possible for the weak learner. Thus, whereas the weak learner on round t aims to find h_t maximizing the weighted accuracy $\Pr_{i \sim D_t}[y_i = h_t(x_i)]$, or equivalently, the weighted correlation

$$\sum_{i=1}^{m} D_t(i) y_i h_t(x_i),$$

the boosting algorithm aims to construct distributions D_t for which this correlation

(or accuracy) will be small for all weak classifiers.

Moreover, on every round t, the new distribution D_{t+1} is constructed so that this correlation with h_t will be zero. This is because, keeping in mind that

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t},$$

it follows that

$$\begin{aligned} \sum_{i=1}^{m} D_{t+1}(i) y_i h_t(x_i) &= \frac{1}{Z_t} \sum_{i=1}^{m} D_t(i) e^{-\alpha_t y_i h_t(x_i)} y_i h_t(x_i) \\ &= -\frac{1}{Z_t} \cdot \frac{dZ_t}{d\alpha_t} = 0 \end{aligned} \tag{8.4}$$

where we regard the normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) e^{-\alpha_t y_i h_t(x_i)}$$

as a function of α_t. Eq. (8.4) follows from the fact that α_t is chosen to minimize Z_t (as shown in Section 3.1), so that $dZ_t/d\alpha_t = 0$.

Thus, it would seem that AdaBoost's aim is to pursue a distribution D for which

$$\sum_{i=1}^{m} D(i) y_i \hbar_j(x_i) = 0 \tag{8.5}$$

for every $\hbar_j \in \mathcal{H}$, that is, a distribution D that is so hard, that no weak classifier $\hbar_j \in \mathcal{H}$ is at all correlated under D with the labels y_i. Eq. (8.5) provides a set of n linear constraints on D corresponding to the constraints in Eq. (8.1). We use these to define the feasible set $\mathcal{P}$; that is, $\mathcal{P}$ is now defined to be the set of all distributions D satisfying *all* these constraints as in Eq. (8.5), for all $\hbar_j \in \mathcal{H}$:

$$\mathcal{P} \doteq \left\{ D : \sum_{i=1}^{m} D(i) y_i \hbar_j(x_i) = 0 \text{ for } j = 1, \ldots, n \right\} \tag{8.6}$$

In Section 8.1.1, we measured distances using the usual Euclidean distance. Now, however, we are working in a space of probability distributions where another measure of distance seems more suitable. In particular, we now find it appropriate to replace Euclidean geometry with the "information geometry" associated with the *relative entropy* distance measure defined between any two distributions as in Eq. (6.8). This change affects how points are projected onto hyperplanes as well as what we mean by the "closest" point in $\mathcal{P}$ to the reference point. Otherwise, the basic algorithm of Figure 8.1 will remain unchanged.

Having chosen the linear constraints defining $\mathcal{P}$, we next need to choose a reference point, analogous to $\mathbf{x}_0$ in Section 8.1.1. In the absence of other information, it seems natural to treat all of the training examples equally, and thus to choose the distribution that is closest to the uniform distribution U over $\{1, \ldots, m\}$.

Putting these ideas together, we arrive at the following optimization program:

$$\begin{aligned} &\text{minimize } \mathrm{RE}\left(D \parallel U\right) \\ &\text{subject to } \sum_{i=1}^{m} D(i) y_i \hbar_j(x_i) = 0 \text{ for } j = 1, \ldots, n. \\ &\qquad D(i) \geq 0 \text{ for } i = 1, \ldots, m; \sum_{i=1}^{m} D(i) = 1. \end{aligned} \tag{8.7}$$

The program has the same form as Program (8.3) except that Euclidean distance has been replaced by relative entropy, and we have also added constraints to make explicit the requirement that D has to be a probability distribution. Note that

$$\mathrm{RE}\left(D \parallel U\right) = \ln m - H(D) \tag{8.8}$$

where $H(D)$ is the *entropy* of D:

$$H(D) = -\sum_{i=1}^{m} D(i) \ln D(i),$$

a measure of the "spread" of the distribution. Thus, minimization of $\mathrm{RE}\left(D \parallel U\right)$ is equivalent to maximization of the entropy $H(D)$, and so this approach also relates strongly to a large and old body of work on *maximum entropy* methods.

To solve Program (8.7), the same iterative projection techniques described in Section 8.1.1 can be applied. Now, however, our method of projection must be modified and must now use relative entropy rather than Euclidean distance. Thus, we begin with $D_1 = U$. Then, on each round, we select a constraint j and compute D_{t+1} to be the projection D_t onto the hyperplane defined by Eq. (8.5). That is, D_{t+1} minimizes $\mathrm{RE}\left(D_{t+1} \parallel D_t\right)$ among all distributions satisfying this equality. The complete algorithm is shown in Figure 8.3.

This algorithm works just like the one in Figure 8.1, except for the change in the distance measure. We begin with the reference point, in this case U, and iteratively project onto individual linear constraints. Like the algorithm of Figure 8.1, this one is known to converge to the (unique) solution of Program (8.7), provided that the feasible set $\mathcal{P}$ is non-empty. In fact, Euclidean distance (squared) and relative entropy are instances of a more general class of distance functions called *Bregman distances* for which it is known that such an iterative projection algorithm will be effective. In the general case, the algorithm is known as *Bregman's algorithm*.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$
$\mathcal{H} = \{\hbar_1, \ldots, \hbar_n\}$
Find: $D \in \mathcal{P}$ minimizing $\mathrm{RE}\,(D \parallel U)$ where $\mathcal{P}$ is as in Eq. (8.6)
$D_1 = U$
For $t = 1, 2, \ldots$ until "close enough"

- choose $h_t \in \mathcal{H}$ defining one of the constraints
- let $D_{t+1} = \arg \min_{D: \sum_{i=1}^m D(i) y_i h_t(x_i) = 0} \mathrm{RE}\,(D \parallel D_t)$

Output D_t

Figure 8.3: An iterative projection algorithm corresponding to AdaBoost.

Let us assume that greedy selection is used in the choice of constraints so that on each round h_t is chosen to maximize the distance to the violated hyperplane, that is, so that

$$\min_{D: \sum_{i=1}^m D(i) y_i h_t(x_i) = 0} \mathrm{RE}\,(D \parallel D_t)$$

will be maximized over all $h_t \in \mathcal{H}$. With this assumption, as already suggested, it can be shown that the algorithm in Figure 8.3 is simply AdaBoost in disguise (with canonical weak hypothesis selection). In other words, the distributions D_t computed by the two algorithms are identical on every round. To see this, suppose $h_t \in \mathcal{H}$ is chosen on round t. Then D_{t+1} is chosen to minimize $\mathrm{RE}\,(D \parallel D_t)$ subject to the constraint

$$\sum_{i=1}^m D(i) y_i h_t(x_i) = 0.$$

We can compute this minimization by forming the Lagrangian

$$\mathcal{L} = \sum_{i=1}^m D(i) \ln \left(\frac{D(i)}{D_t(i)} \right) + \alpha \sum_{i=1}^m D(i) y_i h_t(x_i) + \mu \left(\sum_{i=1}^m D(i) - 1 \right) \tag{8.9}$$

where α and μ are the Lagrange multipliers, and where we have also explicitly taken into account the constraint that

$$\sum_{i=1}^m D(i) = 1. \tag{8.10}$$

Computing derivatives and equating with zero, we get that

$$0 = \frac{\partial \mathcal{L}}{\partial D(i)} = \ln \left(\frac{D(i)}{D_t(i)} \right) + \alpha y_i h_t(x_i) + 1 + \mu.$$

Thus,

$$D(i) = D_t(i)e^{-\alpha y_i h_t(x_i) - 1 - \mu}.$$

Note that μ, an arbitrary constant, will be chosen to enforce Eq. (8.10), giving

$$D(i) = \frac{D_t(i)e^{-\alpha y_i h_t(x_i)}}{Z}$$

where

$$Z = \sum_{i=1}^{m} D_t(i)e^{-\alpha y_i h_t(x_i)}$$

is a normalization factor. Plugging into Eq. (8.9) and simplifying gives

$$\mathcal{L} = -\ln Z.$$

Thus, α will be chosen to maximize $\mathcal{L}$, or equivalently, to minimize Z. This is exactly how α_t is chosen by AdaBoost as noted in Section 3.1. So, identifying D, α and Z with D_{t+1}, α_t and Z_t, we see that the two algorithms behave identically for the same choice of h_t.

(Implicitly, we are permitting AdaBoost to choose a negative value of α_t, which is equivalent to assuming that $\mathcal{H}$ is closed under negation so that $-h \in \mathcal{H}$ whenever $h \in \mathcal{H}$. We also ignore the degenerate case in which AdaBoost selects α_t to be infinite, which is equivalent to $y_i h(x_i)$ being all of the same sign, for some $h \in \mathcal{H}$, in other words, h itself being a perfect classifier for the dataset.)

The choice of h_t will also be the same as in AdaBoost. Continuing the development above, we have that

$$\begin{aligned}
\mathrm{RE}\left(D_{t+1} \parallel D_t\right) &= \sum_{i=1}^{m} D_{t+1}(i)\left(-\alpha_t y_i h_t(x_i) - \ln Z_t\right) \\
&= -\ln Z_t - \alpha_t \sum_{i=1}^{m} D_{t+1} y_i h_t(x_i) \\
&= -\ln Z_t \qquad (8.11)
\end{aligned}$$

where we have used Eq. (8.4). Thus the algorithm in Figure 8.3 chooses h_t to minimize Z_t, which, as shown in Section 7.1, is exactly what AdaBoost does as well.

8.1.3 Conditions for a non-empty feasible set

So we conclude that the two algorithms are identical. Later, in Section 8.2, we present techniques that can be used to show that the distributions D_t of AdaBoost

— that is, the iterative projection algorithm of Figure 8.3 — will converge to a distribution which is the unique solution of Program (8.7) provided the feasible set $\mathcal{P}$ in Eq. (8.6) is non-empty.

This latter condition turns out to be equivalent to the data being linearly inseparable, that is, not linearly separable with positive margin in the sense defined in Section 3.3. This equivalence follows from the equivalence between the weak learning assumption and linear separability, as seen in Section 5.3.3, and from the game-theoretic view of boosting of Section 6.4.1. For if the data is linearly separable with positive margin 2γ, then the γ-weak learning assumption holds, or more specifically, for every distribution D there exists $\hbar_j \in \mathcal{H}$ such that

$$\Pr_{i\sim D}[y_i = \hbar_j(x_i)] \geq \tfrac{1}{2} + \gamma,$$

which is the same as

$$\sum_{i=1}^{m} D(i) y_i \hbar_j(x_i) \geq 2\gamma > 0.$$

This immediately implies that the feasible set $\mathcal{P}$ is empty.

On the other hand, if the data is not linearly separable, then the γ-weak learning assumption does not hold for any $\gamma > 0$; in other words, for every $\gamma > 0$, there is a distribution D such that for all $\hbar_j \in \mathcal{H}$,

$$\Pr_{i\sim D}[y_i = \hbar_j(x_i)] < \tfrac{1}{2} + \gamma.$$

Since we assume that $\mathcal{H}$ is closed under negation, this is the same as

$$\left|\sum_{i=1}^{m} D(i) y_i \hbar_j(x_i)\right| < 2\gamma$$

for all $\hbar_j \in \mathcal{H}$. Thus,

$$\min_{D} \max_{\hbar_j \in \mathcal{H}} \left|\sum_{i=1}^{m} D(i) y_i \hbar_j(x_i)\right| = 0 \tag{8.12}$$

where the minimum over distributions D must exist since the space of all distributions is compact. Eq. (8.12) implies immediately that the minimizing D belongs to $\mathcal{P}$, which therefore cannot be empty.

Thus, linear separability is equivalent both to the weak learning assumption, and the feasible set $\mathcal{P}$ being empty. Moreover, if the data is not linearly separable, then AdaBoost's distributions converge to the unique solution of Program (8.7) as will be seen in Section 8.2. But if the data is linearly separable — which in many respects is the more interesting case since it captures situations in which a

large margin can be attained — then the distributions computed by AdaBoost can never converge to a single distribution since in this case, the distance between D_t and D_{t+1} must be lower bounded by a constant: Using Eq. (8.11), as well as the reasoning (and notation) of Theorem 3.1, particularly Eq. (3.8), we have that

$$\begin{aligned} \mathrm{RE}\left(D_{t+1} \parallel D_t\right) &= -\ln Z_t \\ &= -\tfrac{1}{2}\ln(1-4\gamma_t^2) \\ &\geq -\tfrac{1}{2}\ln(1-4\gamma^2) > 0. \end{aligned}$$

Here, we used the fact that the γ-weak learning assumption holds in this case for some $\gamma > 0$, so that $\gamma_t \geq \gamma$ for all t. Hence, $\mathrm{RE}\left(D_{t+1} \parallel D_t\right)$ is at least some positive constant for every t, so the distributions can never converge.

The behavior of AdaBoost's distributions in the linearly separable case is not fully understood. In all closely examined cases, AdaBoost's distributions have been found to converge eventually to a cycle. However, it is not known if this will always be the case, or if AdaBoost's asymptotic behavior can sometimes be chaotic. The dynamics of these distributions can be quite remarkable. Figure 8.4 shows plots over time of AdaBoost's distributions on just two of the training examples. In some cases, convergence to a tight cycle can be quick; in other cases, the behavior may appear chaotic for some time before the algorithm finally converges to a cycle.

8.1.4 Iterative projection using unnormalized distributions

To alleviate the difficulties posed to the algorithm of Figure 8.3 by linearly separable data, we can shift our attention away from the *normalized* distributions used by AdaBoost and instead consider their form before normalization as *unnormalized* weights on the examples. This will give us an alternative characterization of AdaBoost as an iterative projection algorithm in a way that admits a unified treatment of both the linearly separable and inseparable cases. We will use this formulation to prove the convergence of AdaBoost's (unnormalized) distributions, as well as the convergence of AdaBoost to the minimum of the exponential loss.

So we will now be working with unnormalized weight vectors. Relative entropy, the distance measure we previously used, is no longer appropriate for such non-distributions. However, as noted earlier, there exists a whole family of distance measures with which iterative projection can be used. *Unnormalized* relative entropy is a natural choice. For two non-negative vectors $\mathbf{p}, \mathbf{q} \in \mathbb{R}_+^m$, it is defined to be

$$\mathrm{RE}_u\left(\mathbf{p} \parallel \mathbf{q}\right) = \sum_{i=1}^{m} \left[p_i \ln(p_i/q_i) + q_i - p_i\right].$$

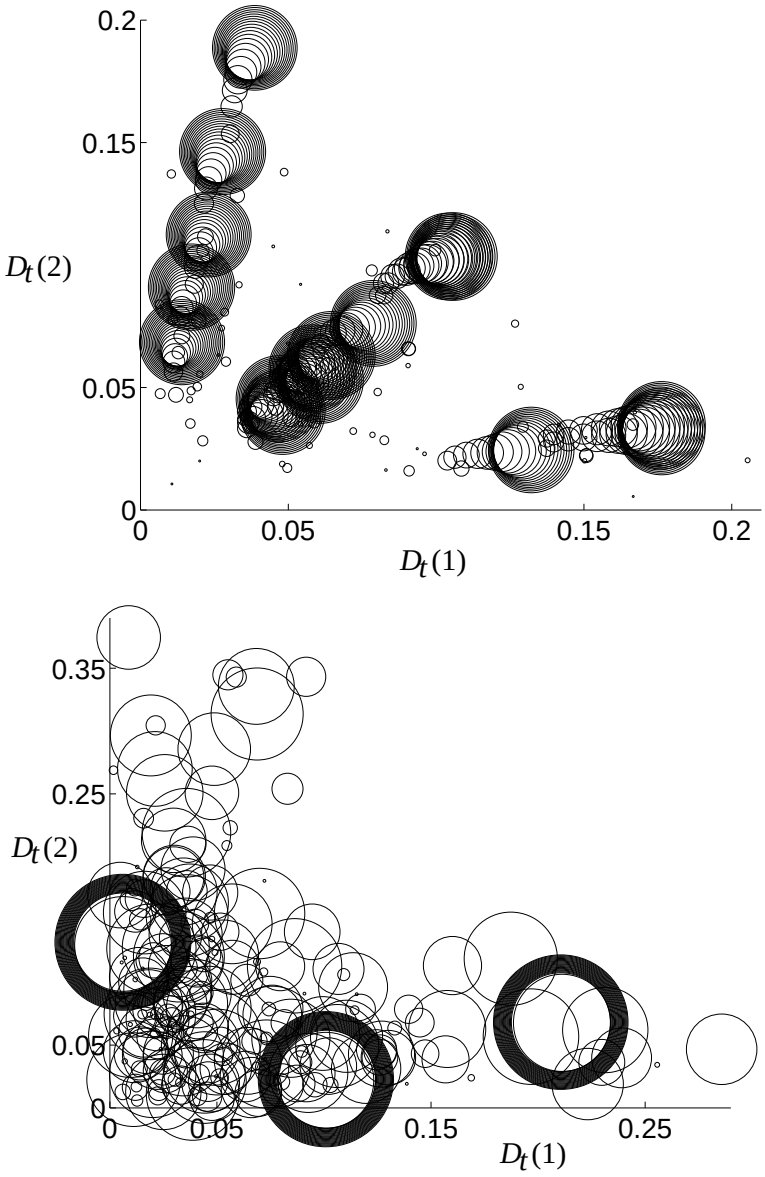

Figure 8.4: Examples of AdaBoost's dynamic behavior on small learning problems. The plots show a projection onto the first two components of the distribution D_t, that is, $D_t(1)$ on the x-axis and $D_t(2)$ on the y-axis. Smaller circles indicate earlier iterations, and larger circles indicate later iterations so that the entire dynamic time course of the algorithm can be observed at once. In both cases, the algorithm is apparently converging to a cycle.

Like standard (normalized) relative entropy, the unnormalized version is always nonnegative, and is equal to zero if and only if $\mathbf{p} = \mathbf{q}$. When clear from context, we write simply RE rather than RE_u.

We can replace relative entropy with unnormalized relative entropy in Program (8.7) and Figure 8.3. We also replace uniform distribution U with the vector $\mathbf{1}$ of all 1's. To emphasize the shift to unnormalized weight vectors, we use lower case $\mathbf{d}$, $\mathbf{d}_t$, etc. to represent unnormalized weight vectors. Thus, the problem now is to find $\mathbf{d}$ so as to solve:

$$\begin{aligned} &\text{minimize } \mathrm{RE}_u\left(\mathbf{d} \parallel \mathbf{1}\right) \\ &\text{subject to } \sum_{i=1}^{m} d_i y_i \hbar_j(x_i) = 0 \text{ for } j = 1, \ldots, n. \\ &\quad d_i \geq 0 \text{ for } i = 1, \ldots, m. \end{aligned} \tag{8.13}$$

We no longer require that $\mathbf{d}$ be normalized. The feasible set associated with this program is

$$\mathcal{P} \doteq \left\{ \mathbf{d} \in \mathbb{R}_+^m : \sum_{i=1}^{m} d_i y_i \hbar_j(x_i) = 0 \text{ for } j = 1, \ldots, n \right\}. \tag{8.14}$$

Earlier, we faced the problem that there might be no distributions satisfying all the constraints. Now that difficulty is entirely erased: the set $\mathcal{P}$ cannot be empty since all the constraints are trivially satisfied when $\mathbf{d} = \mathbf{0}$, the all 0's vector. This is the key advantage of this shift to unnormalized vectors.

Our iterative projection algorithm is nearly unchanged, and is shown for reference in Figure 8.5. Of course, we need to emphasize that the relative entropy used in the figure is *unnormalized* relative entropy, that is, $\mathrm{RE}_u\left(\cdot \parallel \cdot\right)$.

We claim first that this procedure is again equivalent to AdaBoost in the sense that, on every round, the selected h_t will be the same, and the distributions will be in exact correspondence, after normalization, that is,

$$D_t(i) = \frac{d_{t,i}}{\sum_{i=1}^{m} d_{t,i}}.$$

(As before, we assume greedy selection of h_t.) This can be shown exactly as before. For given $h_t \in \mathcal{H}$, $\mathbf{d}_{t+1}$ is selected to minimize $\mathrm{RE}\left(\mathbf{d} \parallel \mathbf{d}_t\right)$ with $\sum_{i=1}^{m} d_i y_i h_t(x_i) = 0$. The Lagrangian now becomes

$$\mathcal{L} = \sum_{i=1}^{m} \left[d_i \ln\left(\frac{d_i}{d_{t,i}}\right) + d_{t,i} - d_i \right] + \alpha \sum_{i=1}^{m} d_i y_i h_t(x_i).$$

Given: $(x_1, y_1), \ldots, (x_m, y_m)$
$\mathcal{H} = \{\hbar_1, \ldots, \hbar_n\}$
Find: $\mathbf{d} \in \mathcal{P}$ minimizing $\mathrm{RE}_u(\mathbf{d} \parallel \mathbf{1})$ where $\mathcal{P}$ is as in Eq. (8.14)
$\mathbf{d}_1 = \mathbf{1}$
For $t = 1, 2, \ldots$ until "close enough"

- choose $h_t \in \mathcal{H}$ defining one of the constraints
- let $\mathbf{d}_{t+1} = \arg \min_{\mathbf{d}: \sum_{i=1}^m d_i y_i h_t(x_i) = 0} \mathrm{RE}_u(\mathbf{d} \parallel \mathbf{d}_t)$

Output $\mathbf{d}_t$

Figure 8.5: An iterative projection algorithm corresponding to AdaBoost using unnormalized relative entropy.

Using calculus to minimize with respect to d_i gives

$$d_i = d_{t,i} e^{-\alpha y_i h_t(x_i)}.$$

Plugging into $\mathcal{L}$ gives

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^m d_{t,i} - \sum_{i=1}^m d_i \\ &= \left(\sum_{i=1}^m d_{t,i} \right) (1 - Z) \end{aligned}$$

where

$$\begin{aligned} Z &= \frac{\sum_{i=1}^m d_{t,i} e^{-\alpha y_i h_t(x_i)}}{\sum_{i=1}^m d_{t,i}} \\ &= \sum_{i=1}^m D_t(i) e^{-\alpha y_i h_t(x_i)}. \end{aligned}$$

So again, α is selected to minimize Z, and again, with $\mathbf{d}$, α and Z identified with $\mathbf{d}_{t+1}$, α_t and Z_t, we see an exact correspondence with the computations of AdaBoost. We also see that the same h_t will be selected on each round by the two algorithms since

$$\begin{aligned} \mathrm{RE}(\mathbf{d}_{t+1} \parallel \mathbf{d}_t) &= -\alpha_t \sum_{i=1}^m d_{t+1,i} y_i h_t(x_i) + \sum_{i=1}^m (d_{t,i} - d_{t+1,i}) \\ &= \left(\sum_{i=1}^m d_{t,i} \right) (1 - Z_t). \end{aligned}$$

So, as for AdaBoost and the algorithm of Figure 8.3, h_t is selected to minimize Z_t. We conclude that all three algorithms are equivalent.

8.2 Proving the convergence of AdaBoost

Now, however, we are in a position to prove some important convergence properties for AdaBoost. We will show that the unnormalized distributions associated with AdaBoost converge to the unique solution of Program (8.13). Further, as a by-product, we will prove that AdaBoost converges asymptotically to the minimum of the exponential loss as discussed, but not proved, in Chapter 7.

To ease notation, let us define an $m \times n$ matrix $\mathbf{M}$ with entries

$$M_{ij} = y_i \hbar_j(x_i).$$

Note that all of the vectors $\mathbf{d}_t$ computed by our algorithm are exponential in a linear combination of the columns of $\mathbf{M}$. That is, all the $\mathbf{d}_t$'s belong to the set $\mathcal{Q}$, where $\mathcal{Q}$ is the set of all vectors $\mathbf{d}$ of the form

$$d_i = \exp\left(-\sum_{j=1}^{n} \lambda_j M_{ij}\right)$$

for $\boldsymbol{\lambda} \in \mathbb{R}^n$. Since the $\mathbf{d}_t$'s all belong to $\mathcal{Q}$, their limit, if it exists, must be in the closure of $\mathcal{Q}$, denoted $\overline{\mathcal{Q}}$. Moreover, the vector we seek must belong to the feasible set $\mathcal{P}$ defined in Eq. (8.14). Thus, informally, it would seem that the algorithm, if effective, will converge to a point in both sets, that is, in $\mathcal{P} \cap \overline{\mathcal{Q}}$. We will prove just such convergence, and we will also prove that this is sufficient for all our purposes.

8.2.1 Two problems in one

We are trying eventually to prove two convergence properties. First, we want to show that the vectors $\mathbf{d}_t$ asymptotically solve Program (8.13). That is, we want to show that their limit is equal to

$$\arg\min_{\mathbf{p}\in\mathcal{P}} \mathrm{RE}\left(\mathbf{p} \parallel \mathbf{1}\right). \tag{8.15}$$

Our second goal is to show that AdaBoost minimizes exponential loss. By unraveling the computation of

$$d_{t+1,i} = d_{t,i} e^{-\alpha_t y_i h_t(x_i)},$$

we have that the sum of these weights after T rounds is

$$\sum_{i=1}^{m} d_{T+1,i} = \sum_{i=1}^{m} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\right)$$

which of course is the exponential loss associated with the combined classifier $\sum_{t=1}^{T} \alpha_t h_t(x)$ as in Section 7.1. More generally, any linear combination of weak classifiers $\sum_{j=1}^{n} \lambda_j \hbar_j(x)$ defines a vector $\mathbf{q} \in \mathcal{Q}$ with

$$q_i = \exp\left(-y_i \sum_{j=1}^{n} \lambda_j \hbar_j(x_i)\right).$$

Indeed, the vectors of $\mathcal{Q}$ are in direct correspondence with linear combinations of weak classifiers. The exponential loss associated with this linear combination is again the sum of the components

$$\sum_{i=1}^{m} q_i = \sum_{i=1}^{m} \exp\left(-y_i \sum_{j=1}^{n} \lambda_j \hbar_j(x_i)\right). \tag{8.16}$$

Thus, our second goal is to show that $\sum_{i=1}^{m} d_{t,i}$ converges to

$$\inf_{\mathbf{q}\in\mathcal{Q}} \sum_{i=1}^{m} q_i = \min_{\mathbf{q}\in\overline{\mathcal{Q}}} \sum_{i=1}^{m} q_i.$$

(The minimum must exist since $\overline{\mathcal{Q}}$ is closed, and since we can restrict our attention, for the purposes of the minimum, to a bounded subset of $\overline{\mathcal{Q}}$, such as only those $\mathbf{q}$ for which $\sum_{i=1}^{m} q_i \leq m$; this restricted set is compact.)

Note that

$$\mathrm{RE}\left(\mathbf{0} \parallel \mathbf{q}\right) = \sum_{i=1}^{m} q_i.$$

Thus, we can summarize that our second goal is to show that the limit of the vectors $\mathbf{d}_t$ is equal to

$$\arg\min_{\mathbf{q}\in\overline{\mathcal{Q}}} \mathrm{RE}\left(\mathbf{0} \parallel \mathbf{q}\right). \tag{8.17}$$

8.2.2 The proof

We show now that AdaBoost converges to a solution of both Eq. (8.15) and Eq. (8.17).

We begin our proof by showing that if $\mathbf{d} \in \mathcal{P} \cap \overline{\mathcal{Q}}$ then a certain equality holds, often referred to as a Pythagorean theorem in analogy to the more standard case that relative entropy is replaced with squared Euclidean distance.

Lemma 8.1 *If $\mathbf{d} \in \mathcal{P} \cap \overline{\mathcal{Q}}$ then for all $\mathbf{p} \in \mathcal{P}$ and for all $\mathbf{q} \in \overline{\mathcal{Q}}$,*

$$\mathrm{RE}\left(\mathbf{p} \parallel \mathbf{q}\right) = \mathrm{RE}\left(\mathbf{p} \parallel \mathbf{d}\right) + \mathrm{RE}\left(\mathbf{d} \parallel \mathbf{q}\right).$$

Proof: We first claim that if $\mathbf{p} \in \mathcal{P}$ and $\mathbf{q} \in \overline{\mathcal{Q}}$, then

$$\sum_{i=1}^{m} p_i \ln q_i = 0.$$

For if $\mathbf{p} \in \mathcal{P}$ and $\mathbf{q} \in \mathcal{Q}$, then there exists $\boldsymbol{\lambda} \in \mathbb{R}^n$ for which

$$q_i = \exp\left(-\sum_{j=1}^{n} \lambda_j M_{ij}\right)$$

for $i = 1, \ldots, m$, and so

$$\begin{aligned}\sum_{i=1}^{m} p_i \ln q_i &= -\sum_{i=1}^{m} p_i \sum_{j=1}^{n} \lambda_j M_{ij} \\ &= -\sum_{j=1}^{n} \lambda_j \sum_{i=1}^{m} p_i M_{ij} = 0\end{aligned}$$

since $\mathbf{p} \in \mathcal{P}$. By continuity, the same holds for $\mathbf{q} \in \overline{\mathcal{Q}}$.

It follows that, for $\mathbf{p} \in \mathcal{P}$ and $\mathbf{q} \in \overline{\mathcal{Q}}$,

$$\begin{aligned}\mathrm{RE}\left(\mathbf{p} \parallel \mathbf{q}\right) &= \sum_{i=1}^{m} \left[p_i \ln p_i - p_i \ln q_i + q_i - p_i\right] \\ &= \sum_{i=1}^{m} \left[p_i \ln p_i + q_i - p_i\right].\end{aligned}$$

Applying to $\mathbf{d} \in \mathcal{P} \cap \overline{\mathcal{Q}}$, $\mathbf{p} \in \mathcal{P}$ and $\mathbf{q} \in \overline{\mathcal{Q}}$ gives

$$\begin{aligned}\mathrm{RE}\left(\mathbf{p} \parallel \mathbf{d}\right) + \mathrm{RE}\left(\mathbf{d} \parallel \mathbf{q}\right) &= \sum_{i=1}^{m} \left[p_i \ln p_i + d_i - p_i\right] + \sum_{i=1}^{m} \left[d_i \ln d_i + q_i - d_i\right] \\ &= \sum_{i=1}^{m} \left[p_i \ln p_i + q_i - p_i\right] \\ &= \mathrm{RE}\left(\mathbf{p} \parallel \mathbf{q}\right)\end{aligned}$$

since $\sum_{i=1}^{m} d_i \ln d_i = 0$ by the claim above ($\mathbf{d}$ being both in $\mathcal{P}$ and $\overline{\mathcal{Q}}$). ■

We next prove the following remarkable characterization of the solutions of Eq. (8.15) and Eq. (8.17): if $\mathbf{d}$ is in $\mathcal{P} \cap \overline{\mathcal{Q}}$, then $\mathbf{d}$ *uniquely* solves *both* of the optimization problems of interest.

Theorem 8.2 *Suppose* $\mathbf{d} \in \mathcal{P} \cap \overline{\mathcal{Q}}$. *Then*

$$\mathbf{d} = \arg\min_{\mathbf{p}\in\mathcal{P}} \mathrm{RE}\left(\mathbf{p} \parallel \mathbf{1}\right)$$

and

$$\mathbf{d} = \arg\min_{\mathbf{q}\in\overline{\mathcal{Q}}} \mathrm{RE}\left(\mathbf{0} \parallel \mathbf{q}\right).$$

Moreover, $\mathbf{d}$ *is the unique minimizer in each case.*

Proof: By Lemma 8.1, since $\mathbf{1} \in \mathcal{Q}$, for any $\mathbf{p} \in \mathcal{P}$,

$$\begin{aligned} \mathrm{RE}\left(\mathbf{p} \parallel \mathbf{1}\right) &= \mathrm{RE}\left(\mathbf{p} \parallel \mathbf{d}\right) + \mathrm{RE}\left(\mathbf{d} \parallel \mathbf{1}\right) \\ &\geq \mathrm{RE}\left(\mathbf{d} \parallel \mathbf{1}\right) \end{aligned}$$

since relative entropy is always nonnegative. Furthermore, $\mathbf{d}$ is the unique minimum since the inequality is strict if $\mathbf{p} \neq \mathbf{d}$. The proof for the other minimization problem is similar. ■

This theorem implies that there can be at most one point at the intersection of $\mathcal{P}$ and $\overline{\mathcal{Q}}$. Of course, however, the theorem does not show that $\mathcal{P} \cap \overline{\mathcal{Q}}$ cannot be empty. Nevertheless, this fact will follow from our analysis below of the iterative projection algorithm which will show that the $\mathbf{d}_t$'s must converge to a point in $\mathcal{P} \cap \overline{\mathcal{Q}}$. Thus, in general, $\mathcal{P} \cap \overline{\mathcal{Q}}$ will always consist of exactly one point to which our algorithm, which is equivalent to AdaBoost, necessarily converges.

Theorem 8.3 *The vectors* $\mathbf{d}_t$ *computed by the iterative projection algorithm of Figure 8.5, or equivalently, the unnormalized weight vectors computed by AdaBoost (with canonical weak hypothesis selection), converge to the unique point* $\mathbf{d}^* \in \mathcal{P} \cap \overline{\mathcal{Q}}$, *and therefore to the unique minimum of* $\mathrm{RE}\left(\mathbf{p} \parallel \mathbf{1}\right)$ *over* $\mathbf{p} \in \mathcal{P}$, *and the unique minimum of* $\mathrm{RE}\left(\mathbf{0} \parallel \mathbf{q}\right)$ *over* $\mathbf{q} \in \overline{\mathcal{Q}}$. *Thus, the exponential loss of the algorithm*

$$\sum_{i=1}^{m} d_{t,i}$$

converges to the minimum possible loss

$$\inf_{\boldsymbol{\lambda}\in\mathbb{R}^n} \sum_{i=1}^{m} \exp\left(-y_i \sum_{j=1}^{n} \lambda_j \hbar_j(x_i)\right).$$

Proof: Based on the foregoing, particularly Theorem 8.2, it suffices to show that the sequence $\mathbf{d}_1, \mathbf{d}_2, \ldots$ converges to a point $\mathbf{d}^*$ in $\mathcal{P} \cap \overline{\mathcal{Q}}$.

Let

$$L_t \doteq \sum_{i=1}^{m} d_{t,i},$$

be the exponential loss at round t. By our choice of h_t and α_t, which, as discussed in Sections 8.1.2 and 8.1.4, are chosen to cause the greatest drop in this loss, we have that

$$L_{t+1} = \min_{j,\alpha} \sum_{i=1}^{m} d_{t,i} e^{-\alpha M_{ij}}. \tag{8.18}$$

In particular, considering the case that $\alpha = 0$, this implies that L_t never increases. By Eq. (8.18), we can regard the difference $L_{t+1} - L_t$ as a function A of the vector $\mathbf{d}_t$:

$$L_{t+1} - L_t = A(\mathbf{d}_t)$$

where

$$A(\mathbf{d}) \doteq \min_{j,\alpha} \sum_{i=1}^{m} d_i e^{-\alpha M_{ij}} - \sum_{i=1}^{m} d_i.$$

Clearly, A is a continuous function. Further, since $L_t \geq 0$ for all t, and since the sequence of L_t's is non-increasing, it follows that the differences $A(\mathbf{d}_t)$ must converge to zero. This fact, together with the next lemma, will help us prove that the limit of the $\mathbf{d}_t$'s is in $\mathcal{P}$.

Lemma 8.4 *If $A(\mathbf{d}) = 0$ then $\mathbf{d} \in \mathcal{P}$.*

Proof: Recall our assumption that $M_{ij} \in \{-1, +1\}$ for all i, j. Let

$$W_j^+ = \sum_{i:M_{ij}=+1} d_i$$

and

$$W_j^- = \sum_{i:M_{ij}=-1} d_i.$$

Then

$$\begin{aligned} \min_{\alpha} \sum_{i=1}^{m} d_i e^{-\alpha M_{ij}} &= \min_{\alpha} (W_j^+ e^{-\alpha} + W_j^- e^{\alpha}) \\ &= 2\sqrt{W_j^+ W_j^-}. \end{aligned}$$

Further, $\sum_{i=1}^{m} d_i = W_j^+ + W_j^-$. Thus,

$$\begin{aligned} A(\mathbf{d}) &= \min_j \left[2\sqrt{W_j^+ W_j^-} - (W_j^+ + W_j^-) \right] \\ &= -\max_j \left(\sqrt{W_j^+} - \sqrt{W_j^-} \right)^2. \end{aligned}$$

If we now suppose that $A(\mathbf{d}) = 0$, then

$$W_j^+ = W_j^-$$

for all j, or equivalently

$$\begin{aligned} 0 = W_j^+ - W_j^- &= \sum_{i:M_{ij}=+1} d_i - \sum_{i:M_{ij}=-1} d_i \\ &= \sum_{i=1}^{m} d_i M_{ij}. \end{aligned}$$

In other words, $\mathbf{d} \in \mathcal{P}$. ■

The $\mathbf{d}_t$ vectors, which cannot have negative components, must all be contained in the compact space $[0, m]^m$, since $0 \leq d_{t,i} \leq \sum_{i=1}^m d_{t,i} \leq \sum_{i=1}^m d_{1,i} = m$. Therefore, there must exist a convergent subsequence $t_1, t_2, \ldots$ with some limit point, call it $\mathbf{d}$; that is,

$$\lim_{k\to\infty} \mathbf{d}_{t_k} = \mathbf{d}.$$

Clearly, each $\mathbf{d}_{t_k} \in \mathcal{Q}$, so the limit $\mathbf{d}$ must be in $\overline{\mathcal{Q}}$. Further, by continuity of A, and our earlier observation that $A(\mathbf{d}_t)$ must converge to zero, we have

$$A(\mathbf{d}) = \lim_{k\to\infty} A(\mathbf{d}_{t_k}) = 0.$$

Thus, by Lemma 8.4, $\mathbf{d} \in \mathcal{P}$. Therefore, $\mathbf{d} \in \mathcal{P} \cap \overline{\mathcal{Q}}$, and indeed, $\mathbf{d}$ must be the unique member $\mathbf{d}^*$ of $\mathcal{P} \cap \overline{\mathcal{Q}}$.

Finally, we claim that the *entire* sequence $\mathbf{d}_1, \mathbf{d}_2, \ldots$ must converge to $\mathbf{d}^*$. Suppose not. Then there exists a neighborhood B of $\mathbf{d}^*$ such that infinitely many of the points in the sequence lie outside of B. This infinite set, being in a compact space, must include a convergent subsequence which, by the argument above, must converge to $\mathbf{d}^*$, the only member of $\mathcal{P} \cap \overline{\mathcal{Q}}$. But this is a contradiction since all the points fall outside the neighborhood B.

Therefore, the entire sequence $\mathbf{d}_1, \mathbf{d}_2, \ldots$ converges to $\mathbf{d}^* \in \mathcal{P} \cap \overline{\mathcal{Q}}$. ■

Theorem 8.3 fully characterizes the convergence properties of the unnormalized weight vectors. However, it tells us what happens to the normalized distributions D_t only when the data is not linearly separable. In this case, the unnormalized vectors $\mathbf{d}_t$ will converge to a unique $\mathbf{d}^* \in \mathcal{P} \cap \overline{\mathcal{Q}}$ which is also the minimum of the exponential loss in Eq. (8.16) by Theorem 8.3. Since the data is linearly inseparable, the exponent in at least one term of this sum must be positive so that the entire sum cannot be less than 1. Thus $\mathbf{d}^*$ must be different than $\mathbf{0}$. This means that the normalized distributions D_t will converge to a unique distribution D^* where

$$D^*(i) = \frac{d_i^*}{\sum_{i=1}^m d_i^*}.$$

On the other hand, when the data is linearly separable, the feasible set $\mathcal{P}$ can consist only of the single point $\mathbf{0}$ (as can be seen, for instance, by applying the reasoning used in Section 8.1.3). Thus, $\mathbf{d}^*$, the only element of $\mathcal{P} \cap \overline{\mathcal{Q}}$, must equal $\mathbf{0}$. So in this case, nothing can be concluded about the convergence of the normalized distributions D_t from the convergence of their unnormalized counterparts, and indeed, we have already discussed the fact that these cannot converge to a single point in this case.

8.2.3 Convex duality

We saw in Section 8.2.2 how the same optimization algorithm can be used apparently to solve two seemingly distinct problems simultaneously. That is, the same algorithm both solves Program (8.13) and also minimizes the exponential loss. On the surface, this might seem like a rather remarkable coincidence. However, at a deeper level, these two problems are not at all unrelated. Rather, it turns out that one is the *convex dual* of the other, a distinct but in some sense equivalent formulation of the same problem.

Starting with Program (8.13), the convex dual can be found by first forming the Lagrangian:

$$\mathcal{L} = \sum_{i=1}^{m} (d_i \ln d_i + 1 - d_i) + \sum_{j=1}^{n} \lambda_j \sum_{i=1}^{m} d_i y_i \hbar_j(x_i)$$

where the λ_j's are the Lagrange multipliers. (As usual, we ignore the constraints $d_i \geq 0$ since these will automatically be satisfied.) Computing partial derivatives and setting to zero yields

$$0 = \frac{\partial \mathcal{L}}{\partial d_i} = \ln d_i + \sum_{j=1}^{n} \lambda_j y_i \hbar_j(x_i)$$

so

$$d_i = \exp\left(-y_i \sum_{j=1}^{n} \lambda_j \hbar_j(x_i) \right).$$

Plugging into $\mathcal{L}$ and simplifying gives

$$\mathcal{L} = m - \sum_{i=1}^{m} \exp\left(-y_i \sum_{j=1}^{n} \lambda_j \hbar_j(x_i) \right).$$

Maximization of $\mathcal{L}$ in the λ_j's is the dual problem, which of course is equivalent to minimization of the exponential loss. In general, the solutions of both the original

"primal" problem and the dual problem occur at the "saddle point" where the Lagrangian $\mathcal{L}$ is minimized in the "primal variables" d_i and maximized in the "dual variables" λ_j.

Thus, it is because of convex duality that the same algorithm for Program (8.13) also minimizes exponential loss.

By a similar calculation, it can be shown that the dual of Program (8.7) is also minimization of exponential loss, or more precisely, maximization of

$$\ln m - \ln \left(\sum_{i=1}^{m} \exp \left(-y_i \sum_{j=1}^{n} \lambda_j \hbar_j(x_i) \right) \right).$$

It is not surprising then that the algorithm of Figure 8.3, which computes exactly the normalized equivalents of the vectors in the algorithm of Figure 8.5, and which also is equivalent to AdaBoost, is a procedure for minimizing the exponential loss as well.

8.3 Unification with logistic regression

In Section 7.4, we studied logistic regression, an approach that we saw is closely related to the exponential-loss minimization employed by AdaBoost. We will now build on our understanding of this close relationship using the framework just developed. We will see that the optimization problem solved by AdaBoost is only a slight variant of the one solved by logistic regression in which a single minimization constraint is removed.

For starters, we observe that by slightly changing the distance function used in convex programs (8.7) and (8.13), we arrive at a different convex program that is equivalent via convex duality to logistic regression. In particular, rather than normalized or unnormalized relative entropy, we can use a form of binary relative entropy, specifically,

$$\mathrm{RE}_b\left(\mathbf{p} \parallel \mathbf{q}\right) = \sum_{i=1}^{m} \left[p_i \ln \left(\frac{p_i}{q_i} \right) + (1 - p_i) \ln \left(\frac{1 - p_i}{1 - q_i} \right) \right]$$

where $\mathbf{p}$ and $\mathbf{q}$ must be in $[0,1]^m$. We also change our reference vector from $\mathbf{1}$ to $(1/2)\mathbf{1}$. The problem now is to find $\mathbf{d} \in [0,1]^m$ to solve the following:

$$\begin{aligned} &\text{minimize } \mathrm{RE}_b\left(\mathbf{d} \parallel (1/2)\mathbf{1}\right) \\ &\text{subject to } \sum_{i=1}^{m} d_i y_i \hbar_j(x_i) = 0 \text{ for } j = 1, \ldots, n \\ &\qquad 0 \le d_i \le 1 \text{ for } i = 1, \ldots, m. \end{aligned} \tag{8.19}$$

By the same sort of calculation as in Section 8.2.3, we find that the dual of this problem is to maximize

$$m \ln 2 - \sum_{i=1}^{m} \ln \left(1 + \exp \left(-y_i \sum_{j=1}^{n} \lambda_j \hbar_j(x_i) \right) \right),$$

or equivalently, to minimize the logistic loss of Eq. (7.20) as studied in Section 7.4. An iterative projection algorithm like in Figures 8.3 and 8.5 might easily be derived for Program (8.19) and thus for logistic regression. In fact, the algorithm AdaBoost.L of Figure 7.6 is an approximate, more analytically tractable version of the algorithm that would so be derived, and indeed the convergence of this algorithm to the minimum of the logistic loss can be proved using the same techniques presented in Section 8.2.

So we see that exponential and logistic loss can be treated in a unified convex-programming framework. Their corresponding programs (8.13) and (8.19) differ only in the choice of distance measure. In fact, these programs can be manipulated further to make the resemblance even more striking. This formulation also addresses the estimation of conditional probabilities as discussed in Section 7.5.

For notational convenience, let us regard each $\hbar_j$ as a function of both x and y where we define $\hbar_j(x, y) = y\hbar_j(x)$. The empirical average of $\hbar_j$ is of course

$$\frac{1}{m} \sum_{i=1}^{m} \hbar_j(x_i, y_i). \tag{8.20}$$

Now suppose $p(y|x)$ is the conditional probability that example x receives label y. Imagine an experiment in which x_i is chosen randomly according to its empirical probability (i.e., uniformly at random from the sample), but then a label y is selected at random according to the true conditional probability distribution $p(\cdot|x_i)$. The expected value of $\hbar_j$ under this "semi-empirical" distribution on pairs (x_i, y) will be

$$\frac{1}{m} \sum_{i=1}^{m} \sum_{y} p(y|x_i)\hbar_j(x_i, y). \tag{8.21}$$

Given enough data, we expect Eqs. (8.20) and (8.21) to be roughly equal since the two are equal in expectation. It therefore may seem natural, in computing an estimate of $p(y|x)$, that we require equality:

$$\frac{1}{m} \sum_{i=1}^{m} \hbar_j(x_i, y_i) = \frac{1}{m} \sum_{i=1}^{m} \sum_{y} p(y|x_i)\hbar_j(x_i, y). \tag{8.22}$$

Naturally, being a conditional probability distribution,

$$\sum_{y} p(y|x) = 1 \tag{8.23}$$

for each x. However, for the purposes of estimating p, we might, in a perverse move, allow this constraint to be dropped. In this case, the left hand side of Eq. (8.22) needs to be adjusted to balance the varying weight on different examples x_i. This gives the requirement that

$$\frac{1}{m}\sum_{i=1}^{m}\hbar_j(x_i,y_i)\left(\sum_y p(y|x_i)\right)=\frac{1}{m}\sum_{i=1}^{m}\sum_y p(y|x_i)\hbar_j(x_i,y). \tag{8.24}$$

So our goal will be to find a set of numbers $p(y|x)$ satisfying Eq. (8.24); in other words, we will now regard these as unknown variables to be solved for, rather than true conditional probabilities. Analogous to the maximum-entropy approach taken earlier in this chapter, among all such sets of numbers satisfying the constraints in Eq. (8.24), we will choose the one that, on average, gives conditional distributions over the labels that are closest to uniform since, a priori, no label should be favored over any other. Moreover, since p might not be normalized, a form of unnormalized relative entropy must be used.

Putting these ideas together yields the following program:

$$\begin{aligned}&\text{minimize } \sum_{i=1}^{m}\mathrm{RE}_u\left(p(\cdot|x_i)\ \|\ \mathbf{1}\right)\\&\text{subject to } \frac{1}{m}\sum_{i=1}^{m}\hbar_j(x_i,y_i)\left(\sum_y p(y|x_i)\right)=\frac{1}{m}\sum_{i=1}^{m}\sum_y p(y|x_i)\hbar_j(x_i,y) \text{ for } j=1,\ldots,n.\\&p(y|x_i)\geq 0 \text{ for } i=1,\ldots,m, \text{ and for all } y\end{aligned} \tag{8.25}$$

where

$$\mathrm{RE}_u\left(p(\cdot|x_i)\ \|\ \mathbf{1}\right)=\sum_y\left[p(y|x)\ln p(y|x)+1-p(y|x)\right].$$

For simplicity, in this section, let us assume that the same example x never appears in the dataset with different labels. Then it can be shown that this program is equivalent to Program (8.13). The correspondence is made by setting $d_i=p(-y_i|x_i)$; the variables $p(y_i|x_i)$ turn out to always equal 1 in the solution, and so are irrelevant. Thus, Program (8.25) minimizes exponential loss.

Suppose now that we add to Program (8.25) a normalization constraint as in Eq. (8.23). The new program is

$$\begin{aligned}&\text{minimize } \sum_{i=1}^{m}\mathrm{RE}_u\left(p(\cdot|x_i)\ \|\ \mathbf{1}\right)\\&\text{subject to } \frac{1}{m}\sum_{i=1}^{m}\hbar_j(x_i,y_i)\left(\sum_y p(y|x_i)\right)=\frac{1}{m}\sum_{i=1}^{m}\sum_y p(y|x_i)\hbar_j(x_i,y) \text{ for } j=1,\ldots,n.\end{aligned}$$

$$p(y|x_i) \geq 0 \text{ for } i = 1, \ldots, m, \text{ and for all } y$$
$$\sum_y p(y|x_i) = 1 \text{ for } i = 1, \ldots, m. \tag{8.26}$$

With this new constraint, it turns out that this program becomes equivalent to Program (8.19) for logistic regression. As before, the correspondence can be seen by setting $d_i = p(-y_i|x_i)$, which implies $p(y_i|x_i) = 1 - d_i$ in this case.

Thus, from this perspective, AdaBoost and logistic regression solve identical optimization problems except that AdaBoost disregards a single normalization constraint.

As for estimation of conditional probabilities as discussed in Section 7.5, it can be verified that for logistic regression, the values $p(y|x)$ associated with Program (8.26) are consistent with those discussed in that section. Further, the estimates $p(y|x)$ for Program (8.25), which omits the normalization constraint, also are consistent with those obtained from AdaBoost in Section 7.5.

8.4 Application to species distribution modeling

As an example of how these ideas can be applied, we consider the problem of modeling the geographic distribution of a given animal or plant species. This is a critical problem in conservation biology: to save a threatened species, one first needs to know where the species prefers to live, and what its requirements are for survival, that is, its ecological "niche." As will be seen shortly, such models have important applications, such as in the design of conservation reserves.

The data available for this problem typically consists of a list of geographic coordinates where the species has been observed, such as the set of locations at the top of Figure 8.6. In addition, there is data on a number of environmental variables, such as average temperature, average rainfall, elevation, etc., which have been measured or estimated across a geographic region of interest. Examples are shown in the middle of Figure 8.6. The goal is to predict which areas within the region satisfy the requirements of the species' ecological niche, that is, where conditions are suitable for the survival of the species. Figure 8.6 at the bottom shows such a map produced using the method described below.

It is often the case that only *presence data* is available indicating the occurrence of the species. Museum and herbarium collections constitute the richest source of occurrence localities, but their collections typically have no information about the *failure* to observe the species at any given location. In addition, many locations have not been surveyed at all. This means that we have *only* positive examples and no negative examples from which to learn. Moreover, the number of sightings

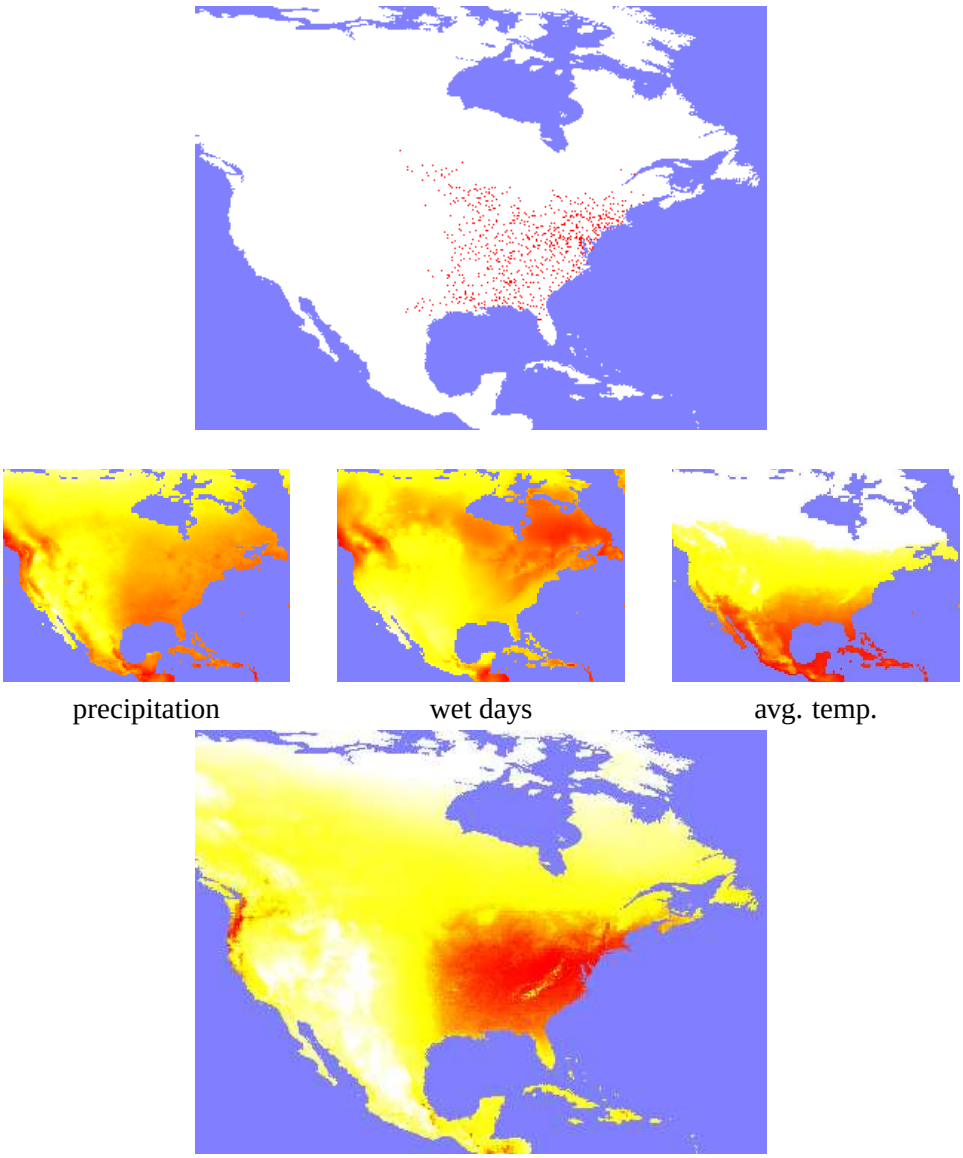

Figure 8.6: Top: a map of the localities where the yellow-throated vireo has been observed. Middle: some sample environmental variables (with darker areas representing higher values). Bottom: the predicted distribution of the species (darker areas indicating locations most suitable for the species).

(training examples) will often be very small by machine-learning standards, say a hundred or less.

Throughout this book, we have focused on the problem of discriminating positive and negative examples. Now, because we only have access to positive examples, we need to take a different approach. Specifically, in modeling the problem, we will assume that the presence records where the species has been observed are being chosen randomly from a probability distribution representing the entire population of the species. Our goal then is to estimate this distribution based on samples randomly chosen from it. In other words, we treat the problem as one of *density estimation*.

More formally, let $\mathcal{X}$ be the large but finite space we are working over, namely, the set of locations on some discretized map of interest. Let Π be a probability distribution over $\mathcal{X}$ representing the distribution of the species across the map. We assume we are given a set of sample locations $x_1, \ldots, x_m$ from $\mathcal{X}$, that is, the observed presence records, which we assume have each been chosen independently at random from Π.

Finally, we are given a set of *base functions* (sometimes also called *features* in this context), $\hbar_1, \ldots, \hbar_n$, which play an analogous role to weak hypotheses in boosting. Each base function $\hbar_j$ provides real-valued information about every point on the map. That is, $\hbar_j : \mathcal{X} \to \mathbb{R}$. For instance, a base function might simply be equal to one of the environmental variables discussed above (such as average temperature). But more generally, it might instead be derived from one or more of these. For example, a base function might be equal to the *square* of an environmental variable (such as elevation squared), or to the *product* of two environment variables (such as elevation times average rainfall). Or, analogous to decision stumps, we might take thresholds of an environmental variable (for instance, 1 if elevation is above 1000 meters, 0 otherwise). In this way, even beginning with a fairly small number of environmental variables, the number of base functions may quickly become quite large.

In all cases, for simplicity, we assume without loss of generality that all base functions have been scaled to have range $[0, 1]$.

Given samples and base functions, the goal is to find a distribution P over $\mathcal{X}$ that is a good estimate of Π. Such an estimate can be interpreted as approximating a measure of the suitability of every location on the map as habitat for the species.

Our approach will be to construct a convex program of a similar form to those studied throughout this chapter, whose solution can be used as such an estimate. Let $\tilde{\Pi}$ denote the empirical distribution over $\mathcal{X}$ that places probability $1/m$ on each of the m samples x_i. A first idea is simply to use $\tilde{\Pi}$ as an estimate of Π. However, this is unlikely to work well since we expect m to be much smaller than $\mathcal{X}$ so that nearly all points in $\mathcal{X}$ will be assigned zero probability mass. Nevertheless, even

though the empirical distribution is a very poor estimate of the true distribution, the empirical average of any base function $\hbar_j$ is likely to be quite a good estimate of its true expectation. That is, we expect

$$\mathrm{E}_{\tilde{\Pi}}[\hbar_j] \approx \mathrm{E}_{\Pi}[\hbar_j]$$

where $\mathrm{E}_{\Pi}[\cdot]$ denotes expectation with respect to the true distribution Π, and similarly $\mathrm{E}_{\tilde{\Pi}}[\cdot]$ denotes empirical average. In fact, using Hoeffding's inequality (Theorem 2.1) and the union bound, we can compute a value of β (roughly $O(\sqrt{(\ln n)/m})$), such that, with high probability,

$$\left|\mathrm{E}_{\Pi}[\hbar_j] - \mathrm{E}_{\tilde{\Pi}}[\hbar_j]\right| \leq \beta \tag{8.27}$$

for all base functions $\hbar_j$. It makes sense then, in constructing P, to ensure that it too satisfies Eq. (8.27), that is, that it belongs to the feasible set

$$\mathcal{P} \doteq \left\{P : \left|\mathrm{E}_P[\hbar_j] - \mathrm{E}_{\tilde{\Pi}}[\hbar_j]\right| \leq \beta \text{ for } j = 1, \ldots, n\right\}.$$

Note that these constraints are linear in the values of P since they can each be rewritten as the two inequalities

$$-\beta \leq \sum_{x \in \mathcal{X}} P(x)\hbar_j(x) - \mathrm{E}_{\tilde{\Pi}}[\hbar_j] \leq \beta.$$

Moreover, $\mathcal{P}$ cannot be empty since it always contains $\tilde{\Pi}$ as a member.

Of the many distributions in $\mathcal{P}$, which one should we pick as our estimate of Π? In the absence of data or other information, it seems natural to assume that all the locations on the map are equally likely to be suitable habitat, in other words, that the uniform distribution U over $\mathcal{X}$ is most reasonable, a priori. This suggests that, among all distributions in $\mathcal{P}$, we choose the one that is closest to U. If using relative entropy as a distance measure as before, then we seek to minimize $\mathrm{RE}(P \parallel U)$, which, from Eq. (8.8), is equivalent to maximizing $H(P)$, the entropy or spread of the distribution P.

Pulling these ideas together, we are proposing estimating Π by selecting the distribution P in $\mathcal{P}$ that has highest entropy, or equivalently, that is closest to uniform in relative entropy. This results in the following optimization problem:

$$\begin{aligned} &\text{minimize } \mathrm{RE}(P \parallel U) \\ &\text{subject to } \left|\mathrm{E}_P[\hbar_j] - \mathrm{E}_{\tilde{\Pi}}[\hbar_j]\right| \leq \beta \text{ for } j = 1, \ldots, n. \\ &\qquad P(x) \geq 0 \text{ for } x \in \mathcal{X}; \sum_{x \in \mathcal{X}} P(x) = 1. \end{aligned} \tag{8.28}$$

This program is nearly of the same form as Program (8.7) except that the linear constraints are somewhat more complicated, involving inequalities rather than equalities. Still, most of the techniques we have discussed generalize immediately to this case.

At this point, we could adapt the iterative projection algorithm of Figure 8.3 (so that, on each iteration, we project onto one of the *half-spaces* associated with the constraints, rather than a hyperplane). Alternatively, using the techniques of Section 8.2.3, it can be shown that the solution of Program (8.28) must be of the form

$$Q_{\boldsymbol{\lambda}}(x) = \frac{1}{Z_{\boldsymbol{\lambda}}} \cdot \exp\left(\sum_{j=1}^{n} \lambda_j \hbar_j(x)\right)$$

for some setting of the parameters $\boldsymbol{\lambda} = \langle \lambda_1, \ldots, \lambda_n \rangle$, where $Z_{\boldsymbol{\lambda}}$ is a normalization factor. In other words, the solution distribution must be proportional to an exponential in some linear combination of the base functions. The convex dual of Program (8.28) turns out to be the problem of finding $\boldsymbol{\lambda}$ which minimizes

$$-\sum_{i=1}^{m} \ln Q_{\boldsymbol{\lambda}}(x_i) + \beta \|\boldsymbol{\lambda}\|_1, \tag{8.29}$$

that is, which minimizes the negative log likelihood of the data (the term on the left), plus a penalty or regularization term (on the right) that has the effect of limiting the size of the weights λ_j on the base functions.

So to solve Program (8.28), we only need to find $\boldsymbol{\lambda}$ minimizing Eq. (8.29). Even for a very large set of base functions, this can often be done efficiently using the general techniques described in Chapter 7 which adjust one parameter at a time to greedily minimize the objective function. That is, although we do not give the details, it is possible to derive a boosting-like algorithm that, on each of a series of rounds, greedily chooses one base function $\hbar_j$ whose associated weight λ_j is adjusted by some value α so as to (approximately) cause the greatest drop in Eq. (8.29).

This *maxent* approach to species distribution modeling has been used and tested on a wide range of datasets. In one large-scale study, it was compared to fifteen other methods on some 226 plant and animal species from six world regions. The median dataset had less than 60 presence records. Maxent, on average, performed better than all other methods except for one based on boosted regression trees (see Section 7.3.2) which was slightly better.

Maxent was also used as part of a large 2008 study of reserve design in Madagascar. This island nation off the southeastern coast of Africa is a biological "hot spot," one of a small number of areas which together cover just 2.3% of the earth's

land surface, but where half of all plant and three-quarters of all vertebrate species are concentrated.

In 2003, the government of Madagascar announced a commitment to triple protected land areas from 2.9% of the island to 10%. By 2006, protected areas had already expanded to 6.3%, but an opportunity existed to carefully select the remaining 3.7%, while also evaluating the design decisions made up to that point.

For this purpose, data was gathered together on some 2,315 species from six taxonomic groups. Maxent was then applied to build a distributional model for all species with at least eight presence records. Finally, with a model in hand for each species, a proposed reserve could be constructed using an algorithm called "Zonation" with the purpose of identifying the areas most suitable to the most species based on such models.

The study found that the existing protected areas, covering 6.3% of the island, actually were rather deficient, entirely omitting 28% of the species studied, meaning that these areas were not protecting any significant part of their habitat. It was further found that an alternative design protecting an equivalent amount of land would have protected *all* of the species.

Clearly, of course, it is too late to significantly modify land areas that have already been set aside. Fortunately, however, the study found that it would still be possible to add on to existing conservation areas in a way that protects all species without exceeding the government's target of designating 10% of the land for conservation. These actual and proposed areas are shown on the map in Figure 8.7.

This study was able to successfully provide detailed recommendations in large part because of the great number of species modeled, and because of the high-resolution models that are possible with maxent.

Summary

In summary, we have provided another powerful perspective on AdaBoost which turns out to be a special case of a family of iterative projection algorithms. This view provides geometric intuitions, as well as the necessary tools to prove fundamental convergence properties of the algorithm.

With respect to convergence, we have seen that there are two basic cases (assuming canonical weak hypothesis selection): If the data is linearly separable, which is equivalent to the weak learning assumption holding, then the distributions D_t computed by AdaBoost cannot converge, but the exponential loss will converge to zero. If the data is not linearly separable, then the distributions D_t will converge to the unique solution of a certain convex program, and the weak edges will converge to zero. In both cases, the exponential loss is asymptotically minimized.

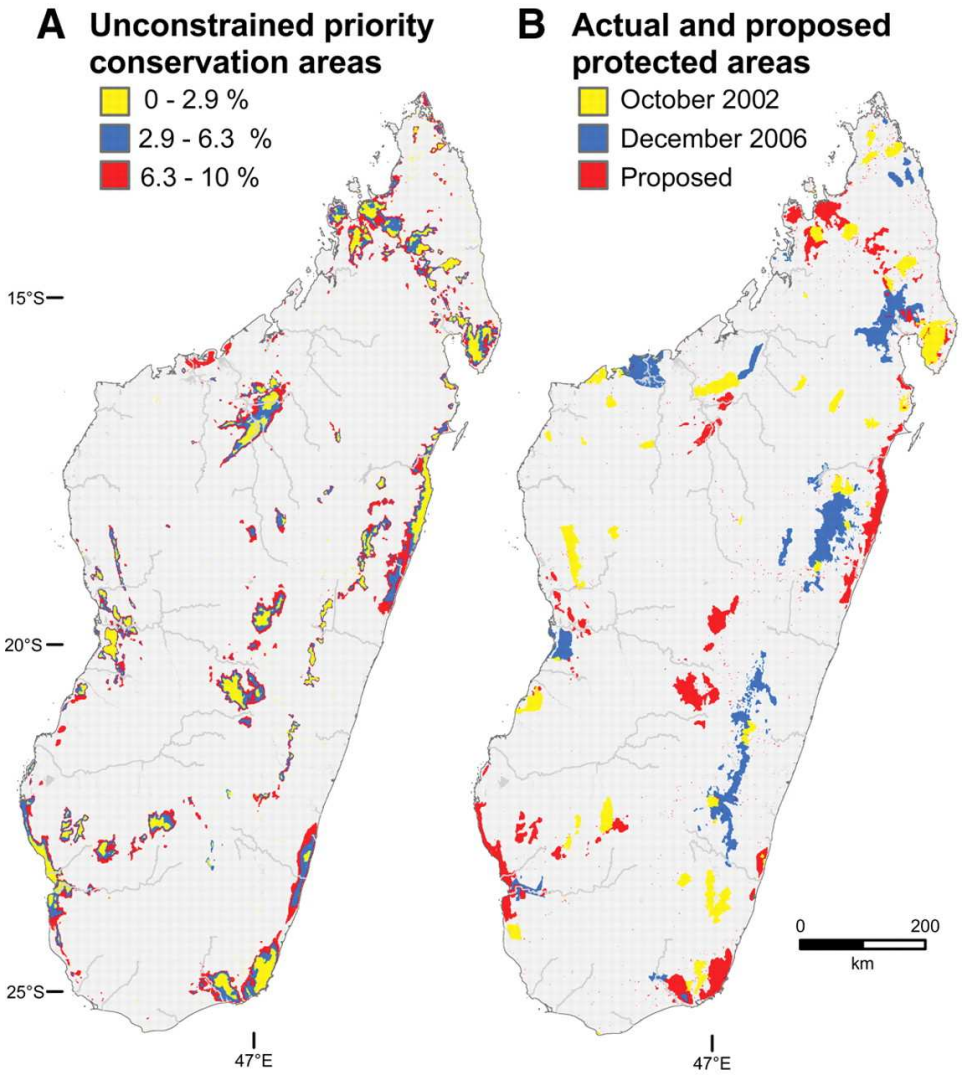

Figure 8.7: On the left, a prioritized map of proposed areas for conservation in Madagascar, unconstrained by previous design decisions. The map shows the top 2.9% of the land that would be chosen as reserve (equivalent in area to what was actually protected in 2002), followed by the next 3.4% (giving an area equivalent to the 6.3% protected in 2006), and finally the last 3.7%, giving total protected area of 10%. On the right, a map showing the actual 2.9% protected in 2002, and the actual additional 3.4% protected through 2006, plus a final proposed expansion of protected areas by 3.7% which would protect all species in the study.

In addition, this approach provides further unification with logistic regression, showing that convex programs associated with AdaBoost and logistic regression differ only in a single constraint.

Finally, we saw how the ideas in this chapter could be applied for the general problem of density estimation, and specifically for modeling the habitat of plant and animal species.

Bibliographic notes

censor and zenios

bregman

kivinen & warmuth

jaynes, etc

dynamics of boosting papers

della pietra

bauschke & borwein

check out references for inconsistent case in censor papers, e.g., "On the behavior of subgradient projections methods for convex feasibility problems in Euclidean spaces" – see ref's [5] & [34]

dudik

kremen et al

maxent in general