Assignment 5

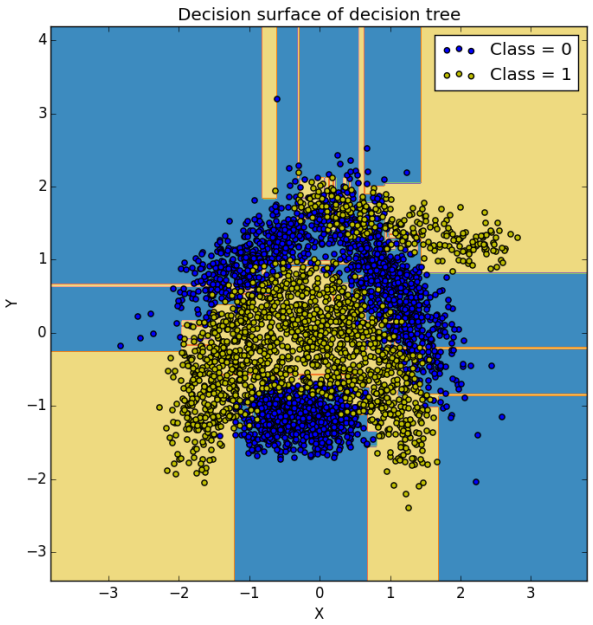
Benjamin Jakubowski

April 5, 2016

2. Decision Trees

2.1 Visualizing Banana dataset decision tree

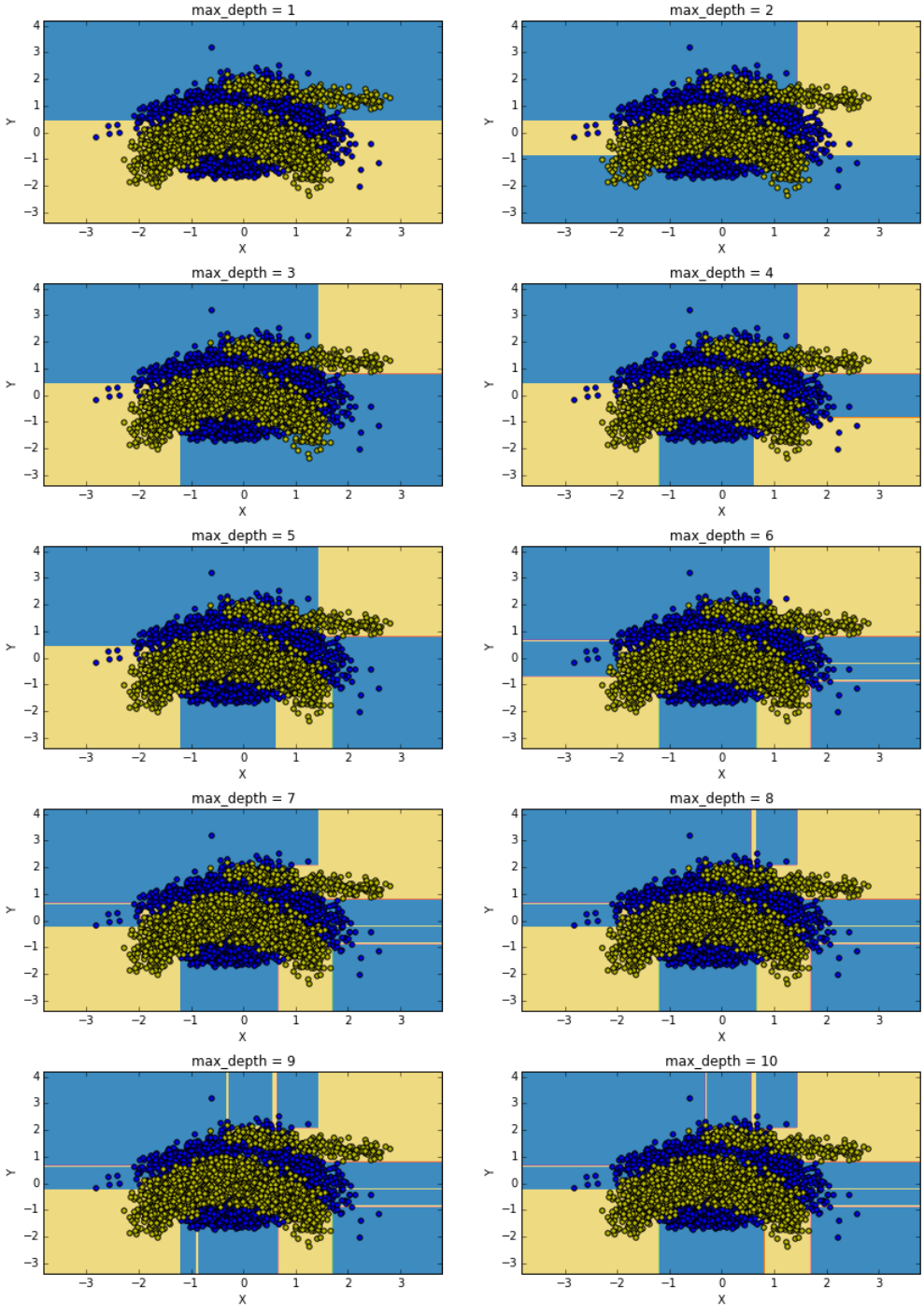
Recall that our objective is to grow and visualize an unpruned decision tree for the "Banana" dataset. The decision surface for this tree is shown below:



Code to reproduce this plot is included in the appendix.

2.2 Visualizing decision surfaces for trees varying `max_depth`

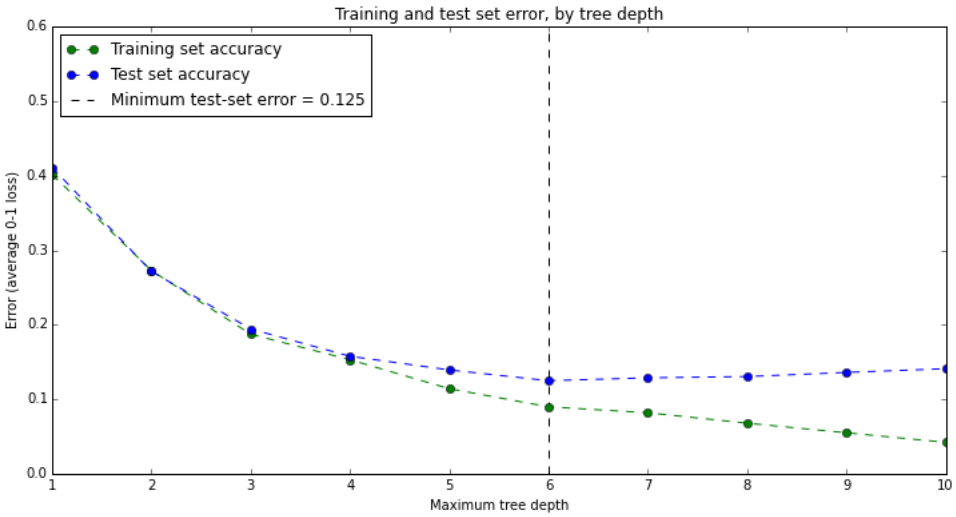
Next, we grew decision trees varying the hyperparameter `max_depth`. Trees were grown for all `max_depth`s in $\{1, \cdots, 10\}$; decision surfaces for these trees are shown below.


max_depth = 1
max_depth = 2
max_depth = 3
max_depth = 4
max_depth = 5
max_depth = 6
max_depth = 7
max_depth = 8
max_depth = 9
max_depth = 10
X
Y

From the decision surfaces, it is apparent the complexity of the trees grow with `max_depth`. Again, code to reproduce these plots is included in the appendix.

2.3 Comparing train and test errors for varying `max_depth`

Next, we compare the training and test set error rates (average 0-1 loss) for trees of varying `max_depth`.



The optimal `max_depth` is 6, though the test set error curve is relatively flat for `max_depth` $\in \{5, \cdots 8\}$. As expected, the training error is strictly decreasing.

2.4 Comparing train and test errors for varying `max_depth`

Using `scikit-learns` `grid_search.GridSearchCV` function, decision trees were build and compared (using cross-validation performance) for all combinations of

- **Criterion**: Gini, entropy
- `max_depth`: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- `min_samples_split`: 2, 4, 8, 16, 32, 64, 128, 256, 512
- `min_samples_leaf`: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512

Based on this grid search, the optimal hyperparameters were found to be

- **Criterion**: Gini
- `max_depth`: 9

- `min_samples_split`: 8
- `min_samples_leaf`: 2

This set of hyperparameters achieved a cross-validation average error rate of 0.8911. However, when this tree was scored using the training set, it only achieved a test-set error rate of 0.137 (which is actually worse than the tree found in 2.1). Hence, we were unable to improve our performance using the hyperparameters tested in our grid search.

3. AdaBoost

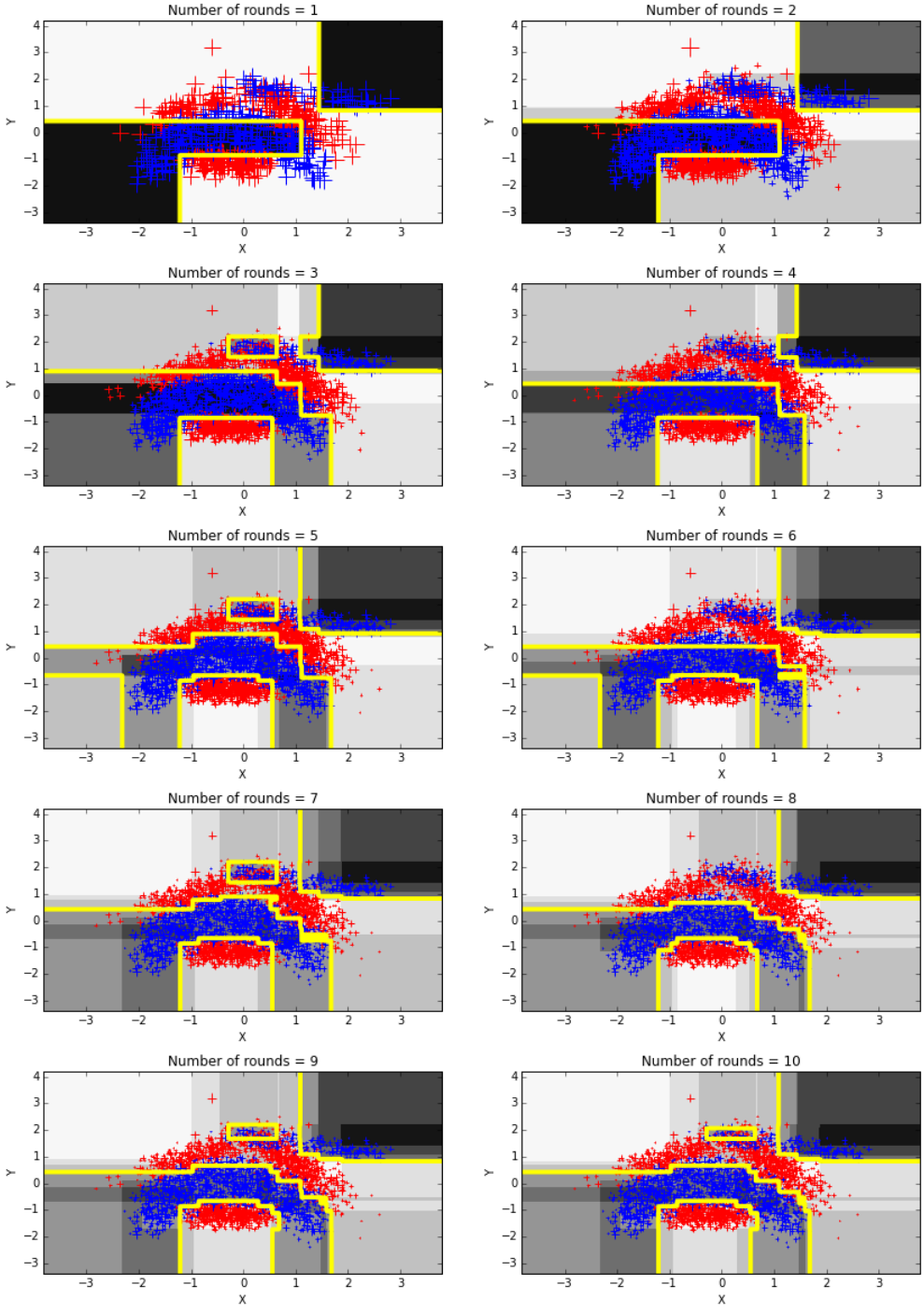
3.1 Implementing AdaBoost

See the code in the attached appendix.

3.2 Visualizing the AdaBoost training procedure

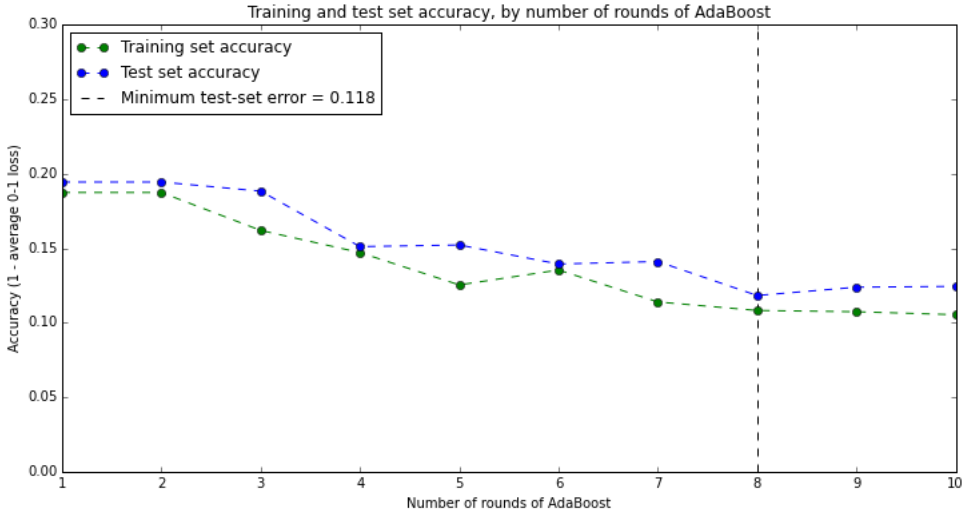
AdaBoost was used to fit the Banana dataset, using decision trees of depth 3 as the base classifiers. The algorithm was run for different numbers of rounds $1, \cdots, 10$, and decision boundaries are shown in the following figure. The size of the markers indicates the final sample weights (though, unfortunately, the tail of the weight distribution is too fat to allow for effective visualization of a small number of influential points).
From these plots, we can make a number of (fairly trivial) observations:

- It is apparent from the shape of the decision surface in the rounds = 1 plot that the base classifier is a tree of depth 3. Note this isn't surprising (since we set this base classifier hyperparameter ourselves)- I was just somewhat surprised since (having forgotten that we set `max_depth` to 3 I had the mistaken assumption most tree boosting algorithms use decision stumps.
- It is apparent the decision surface becomes more complex as we increase the number of rounds. Perhaps more interestingly, modifications to the decision boundary appear to become increasingly marginal (i.e. compare the change from 2 to 3 rounds, to the change from 9 to 10 rounds).
- Again trivial, but worth noting- the decision boundary is decidedly non-linear. This is obvious from the algorithm (since we're using a linear combination of non-linear functions), but worth noting. If we fit a linear model to this data (using only the two features given, without any additional feature engineering or kernelization) we'd do terribly.


Number of rounds = 1
Number of rounds = 2
Number of rounds = 3
Number of rounds = 4
Number of rounds = 5
Number of rounds = 6
Number of rounds = 7
Number of rounds = 8
Number of rounds = 9
Number of rounds = 10
X
Y

3.3 Comparing train and test errors for varying number of rounds

Next, we compare the training and test set error rates (average 0-1 loss) for AdaBoost with varying numbers of rounds.



The minimum test set error is 0.118, for 8 rounds of AdaBoost (though it essentially appear that after 8 rounds, the test set error has leveled out). The training error is still decreasing (as anticipated) after 10 rounds.

4. Gradient Boosting Machines

4.1 Basis functions in L_2-Boosting

In this problem, take $\mathcal{Y} = \mathbb{R}$, and suppose our loss function is given by

$$\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

In the beginning of the m^{th} round of gradient boosting, we have the function $f_{m-1}(x)$. Our goal is to show the h_m chosen as the next basis function is given by

$$h_m = \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} [(y_i - f_{m-1}(x_i)) - h(x_i)]^2$$

First, given $f_{m-1}(x)$, the gradient boosting algorithm proceeds by computing $\mathbf{g}_m$.

$$\begin{aligned}
\mathbf{g}_m &= \left(\frac{\partial}{\partial f(x_i)} \sum_{i=1}^{n} \ell(y_i, f(x_i)) \Big|_{f(x_i)=f_{m-1}(x_i)} \right)_{i=1}^{n} \\
&= \left(\frac{\partial}{\partial f_{m-1}(x_i)} \sum_{i=1}^{n} \ell(y_i, f_{m-1}(x_i)) \right)_{i=1}^{n} \\
&= \left(\frac{\partial}{\partial f_{m-1}(x_i)} \sum_{i=1}^{n} \left[\frac{1}{2} (f_{m-1}(x_i) - y_i)^2 \right] \right)_{i=1}^{n} \\
&= (f_{m-1}(x_i) - y_i)_{i=1}^{n}
\end{aligned}$$

Interestingly (as remarked elsewhere- either a reference text or lecture), $\mathbf{g}_m$ is just the residual vector r_{m-1}.
Now we proceed to fit a regression model to $-\mathbf{g}_m$.

$$\begin{aligned}
h_m &= \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} [(-\mathbf{g}_m)_i - h(x_i)]^2 \\
&= \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} [(-(f_{m-1}(x_i) - y_i)) - h(x_i)]^2 \\
&= \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} [(y_i - f_{m-1}(x_i)) - h(x_i)]^2
\end{aligned}$$

which is what we set out to show.

4.2 Basis functions in BinomialBoost (classification with logistic loss)

Now take $\mathcal{Y} = \{-1, 1\}$, and $\ell(m) = \ln(1 + e^{-m}$, where $m = yf(x)$.
Again, we consider the m^{th} step in gradient boosting, starting with $f_{m-1}(x)$. We again begin by finding $\mathbf{g}_m$.

$$
\begin{aligned}
\mathbf{g}_m &= \left(\frac{\partial}{\partial f(x_i)} \sum_{i=1}^{n} \ell(y_i, f(x_i)) \Big|_{f(x_i)=f_{m-1}(x_i)} \right)_{i=1}^{n} \\
&= \left(\frac{\partial}{\partial f_{m-1}(x_i)} \sum_{i=1}^{n} \ell(y_i, f_{m-1}(x_i)) \right)_{i=1}^{n} \\
&= \left(\frac{\partial}{\partial f_{m-1}(x_i)} \sum_{i=1}^{n} \left[\ln(1 + e^{y_i f_{m-1}(x_i)}) \right] \right)_{i=1}^{n} \\
&= \left(\frac{-y_i}{1 + e^{y_i f_{m-1}(x_i)}} \right)_{i=1}^{n}
\end{aligned}
$$

Finally, we proceed to fit a regression model to $-\mathbf{g}_m$.

$$
\begin{aligned}
h_m &= \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} [(-\mathbf{g}_m)_i - h(x_i)]^2 \\
&= \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left[\left(- \left(\frac{-y_i}{1 + e^{y_i f_{m-1}(x_i)}} \right) \right) - h(x_i) \right]^2 \\
&= \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left[\frac{-y_i}{1 + e^{y_i f_{m-1}(x_i)}} - h(x_i) \right]^2
\end{aligned}
$$

So our expression for h_m is

$$
h_m = \arg\min_{h \in \mathcal{F}} \sum_{i=1}^{n} \left[\frac{-y_i}{1 + e^{y_i f_{m-1}(x_i)}} - h(x_i) \right]^2
$$

5. From Margins to Conditional Probabilities

5.1 Expressing $\mathbb{E}_y[\ell(yf(x))|x]$ in terms of $\pi(x)$ and $\ell(f(x))$

Let $\pi(x) = P(y = 1|x)$. Then note (with $y \in \{1, -1\}$

$$
(1 - \pi(x) = (1 - P(y = 1|x) = P(y = -1|x)
$$

Now consider $\mathbb{E}_y[\ell(yf(x))|x]$. Well,

$$
\begin{aligned}
\mathbb{E}_y[\ell(yf(x))|x] &= \sum_{y \in \{-1,1\}} \ell(yf(x)) P_{Y|X}(y|x) \\
&= \ell(1 \cdot f(x)) P(Y = 1|x) + \ell(-1 \cdot f(x)) P(Y = -1|x) \\
&= \ell(f(x))\pi(x) + \ell(-f(x))(1 - \pi(x))
\end{aligned}
$$

5.2 Bayes prediction function for exponential loss function

Now, let $\ell(y, f(x)) = e^{-yf(x)}$. This is a margin loss, with $m = yf(x)$, so we have $\ell(m) = e^{-m}$.
Now let's find $f^*(x)$. We proceed by differentiating our expression for $\mathbb{E}_y[\ell(yf(x))|x]$ with respect to $f(x)$:

$$\begin{aligned}\frac{\partial}{\partial f(x)}\mathbb{E}_y[\ell(yf(x))|x] &= \frac{\partial}{\partial f(x)}\left[\ell(f(x))\pi(x) + \ell(-f(x))(1-\pi(x))\right] \\ &= \ell'(f(x))f'(x)\pi(x) - \ell'(-f(x))f'(x)(1-\pi(x))\end{aligned}$$

Setting this expression equal to zero, and dividing by $f'(x)$ yields

$$\ell'(f(x))\pi(x) - \ell'(-f(x))(1-\pi(x)) = 0$$

Note $\ell'(m) = -e^{-m}$. Thus, we have

$$\begin{aligned} & -e^{-f(x)}\pi(x) + e^{f(x)}(1-\pi(x)) = 0 \\ \implies \quad & e^{f(x)}\left(-e^{-f(x)}\pi(x) + e^{f(x)}(1-\pi(x))\right) = e^{f(x)}0 \\ \implies \quad & -1\cdot\pi(x) + e^{2f(x)}(1-\pi(x)) = 0 \\ \implies \quad & f(x) = \frac{1}{2}\ln\left(\frac{\pi(x)}{1-\pi(x)}\right)\end{aligned}$$

So $f^*(x) = \frac{1}{2}\ln\left(\frac{\pi(x)}{1-\pi(x)}\right)$.

Next, given $f^*(x)$, note

$$\begin{aligned} & f^*(x) = \frac{1}{2}\ln\left(\frac{\pi(x)}{1-\pi(x)}\right) \\ \implies \quad & 2f^*(x) = \ln\left(\frac{\pi(x)}{1-\pi(x)}\right) \\ \implies \quad & e^{2f^*(x)} = \frac{\pi(x)}{1-\pi(x)} \\ \implies \quad & e^{-2f^*(x)} = \frac{1-\pi(x)}{\pi(x)} \\ \implies \quad & e^{-2f^*(x)} = \frac{1}{\pi(x)} - 1 \\ \implies \quad & 1 + e^{-2f^*(x)} = \frac{1}{\pi(x)} \\ \implies \quad & \frac{1}{1+e^{-2f^*(x)}} = \pi(x)\end{aligned}$$

5.3 Bayes prediction function for logistic loss function

Now, let $\ell(y, f(x)) = \ln(1 + e^{-yf(x)})$. Again this is a margin loss (with $m = yf(x)$), so we have $\ell(m) = \ln(1 + e^{-m})$. Moreover, $\ell'(m) = \frac{-e^{-m}}{1+e^{-m}}$.

Thus, substituting into $\ell'(f(x))\pi(x) - \ell'(-f(x))(1 - \pi(x)) = 0$ yields

$$\begin{aligned}
&\frac{-e^{-f(x)}}{1+e^{-f(x)}}\pi(x) + \frac{e^{f(x)}}{1+e^{f(x)}}(1-\pi(x)) = 0 \\
\implies \quad &\frac{e^{f(x)}}{1+e^{f(x)}}(1-\pi(x)) = \frac{e^{-f(x)}}{1+e^{-f(x)}}\pi(x) \\
\implies \quad &\frac{e^{f(x)}(1+e^{-f(x)})}{e^{-f(x)}(1+e^{f(x)})} = \frac{\pi(x)}{(1-\pi(x))} \\
\implies \quad &\frac{e^{f(x)}e^{f(x)}(1+e^{-f(x)})}{(1+e^{f(x)})} = \frac{\pi(x)}{(1-\pi(x))} \\
\implies \quad &\frac{e^{f(x)}(e^{f(x)}+1)}{(1+e^{f(x)})} = \frac{\pi(x)}{(1-\pi(x))} \\
\implies \quad &e^{f(x)} = \frac{\pi(x)}{(1-\pi(x))} \\
\implies \quad &f(x) = \ln\left(\frac{\pi(x)}{(1-\pi(x))}\right)
\end{aligned}$$

So $f^*(x) = \ln\left(\frac{\pi(x)}{(1-\pi(x))}\right)$.

Next, given $f^*(x)$, note

$$\begin{aligned}
&f^*(x) = \ln\left(\frac{\pi(x)}{(1-\pi(x))}\right) \\
\implies \quad &e^{f^*(x)} = \frac{\pi(x)}{(1-\pi(x))} \\
\implies \quad &e^{-f^*(x)} = \frac{(1-\pi(x))}{\pi(x)} \\
\implies \quad &e^{-f^*(x)} = \frac{1}{\pi(x)} - 1 \\
\implies \quad &1 + e^{-f^*(x)} = \frac{1}{\pi(x)} \\
\implies \quad &\frac{1}{1+e^{-f^*(x)}} = \pi(x)
\end{aligned}$$

5.4 Bayes prediction function for hinge loss function

Now take $\ell(y, f(x)) = \max\{0, 1 - yf(x)\}$.

Again, this is a margin loss, with $\ell(m) = \max\{0, 1 - m\}$.

Recall we want to minimize $\ell(f(x))\pi(x) + \ell(-f(x))(1 - \pi(x))$. Since ℓ is no longer differentiable (over all m), we'll need to use its subdifferential. Our goal is to find $f^*(x)$ that minimizes this expression (i.e. values of $f^*(x)$ where 0 is in the subdifferential).

First, note

$$\partial\left[\ell(f(x))\pi(x) + \ell(-f(x))(1 - \pi(x))\right] = \partial\ell(f(x))\partial f(x)\pi(x) - \partial\ell(-f(x))\partial f(x)(1 - \pi(x))$$

Thus, if

$$0 \in \partial\ell(f(x))\partial f(x)\pi(x) - \partial\ell(-f(x))\partial f(x)(1 - \pi(x))$$

then there exists g's in the ∂'s such that

$$\begin{aligned} & g_{\partial\ell(f(x))}g_{\partial f(x)}\pi(x) - g_{\partial\ell(-f(x))}g_{\partial f(x)}(1 - \pi(x)) = 0 \\ \implies \quad & g_{\partial\ell(f(x))}\pi(x) - g_{\partial\ell(-f(x))}(1 - \pi(x)) = 0 \end{aligned}$$

Now, note

$$\partial\ell(f(x)) = \begin{cases} 0 & \text{for } f(x) > 1 \\ [-1, 0] & \text{for } f(x) = 1 \\ -1 & \text{for } f(x) < 1 \end{cases}$$

and

$$\partial\ell(-f(x)) = \begin{cases} 0 & \text{for } f(x) < -1 \\ [0, 1] & \text{for } f(x) = -1 \\ 1 & \text{for } f(x) > -1 \end{cases}$$

Thus, substituting we can construct

$$\partial\left[\ell(f(x))\pi(x) + \ell(-f(x))(1 - \pi(x))\right] = \begin{cases} (1 - \pi(x)) & \text{for } f(x) > 1 \\ a\pi(x) + (1 - \pi(x)), -1 \le a \le 0 & \text{for } f(x) = 1 \\ -\pi(x) + (1 - \pi(x)) & \text{for } -1 < f(x) < 1 \\ -\pi(x) + b(1 - \pi(x)), 0 \le b \le 1 & \text{for } f(x) = -1 \\ -\pi(x) & \text{for } f(x) < -1 \end{cases}$$

This yields our solution:

- **Case 1**: If $\pi(x) = 1$, then $f^*(x) = c$ for any $c \ge 1$.

- **Case 2**: If $\pi(x) = 0$, then $f^*(x) = c$ for any $c \leq -1$.
- **Case 3**: If $\pi(x) = \frac{1}{2}$, then $f^*(x) = c$ for any $-1 \leq c \leq 1$.
- **Case 4**: If $\pi(x) < \frac{1}{2}$, then $f^*(x) = -1$. To see that $0 \in \partial$ at $f^*(x) = -1$, take $b = \frac{\pi(x)}{1-\pi(x)}$ (which is clearly in $[0, 1]$). Then
  $-\pi(x) + b(1 - \pi(x)) = -\pi(x) + \frac{\pi(x)}{1-\pi(x)}(1 - \pi(x)) = -\pi(x) + \pi(x) = 0$
- **Case 5**: If $\pi(x) > \frac{1}{2}$, then $f^*(x) = 1$. To see $0 \in \partial$ at $f^*(x) = 1$, take $a = \frac{\pi(x)-1}{\pi(x)}$ (which is clearly in $[-1, 0]$). Then
  $a\pi(x) + (1 - \pi(x)) = \frac{\pi(x)-1}{\pi(x)}\pi(x) + (1 - \pi(x)) = \pi(x) - 1 + (1 - \pi(x)) = 0$

We can summarize these cases quite simply by stating:

$$f^*(x) = \text{sign}\left(\pi(x) - \frac{1}{2}\right)$$

6. AdaBoost Actually Works- Exponential Bound on Training Loss

Our objective in this section is to place an exponential bound on the AdaBoost training loss. We do so by proving a number of inequalities, then putting them together in subsection 6.

6.1 $1(g(x) \neq y) < \exp(-yg(x))$

We start be showing for any function g into {-1, +1}, $1(g(x) \neq y) < \exp(-yg(x))$. We show this explicitly (through enumeration):

$g(x)$	y	$1(g(x) \neq y)$	$\exp(-yg(x))$
-1	-1	0	$\exp(-(-1)(-1)) = e^{-1}$
-1	1	1	$\exp(-(-1)(1)) = e$
1	-1	1	$\exp(-(1)(-1)) = e$
1	1	0	$\exp(-(1)(1)) = e^{-1}$

6.2 $L(G, D) < Z_T$

Now we show $L(G, D) < Z_T$. First, recall

$$L(G, D) = \frac{1}{n}\sum_{i=1}^{n} 1(G(x_i) \neq y_i)$$

Now let's consider two cases:

- For i such that $G(x_i) \neq y_i$, we have

$$1(G(x_i) \neq y_i) = 1$$

while

$$\exp(-y_i f_T(x_i)) = \exp(-y_i \sum_{t=1}^{T} \alpha_t G_t(x_i))$$

By $G(x_i) \neq y_i$, the definition of G implies $-y_i \sum_{t=1}^{T} \alpha_t G_t(x_i) \geq 0$, so

$$\exp(-y_i f_T(x_i)) \geq 1 = 1(G(x_i) \neq y_i)$$

- For i such that $G(x_i) = y_i$, we have

$$1(G(x_i) \neq y_i) = 0$$

while $\exp(-y_i f_T(x_i)) > 0$, so

$$\exp(-y_i f_T(x_i)) > 0 = 1(G(x_i) \neq y_i)$$

Thus, matching term by term yields

$$L(G, D) = \frac{1}{n} \sum_{i=1}^{n} 1(G(x_i) \neq y_i) < \frac{1}{n} \sum_{i=1}^{n} \exp(-y_i f_T(x_i)) = Z_T$$

6.3 $w_i^{t+1} < \exp(-y_i f_t(x_i))$

We take a slightly modified approach here, and show $w_i^{t+1} < \frac{1}{n} \exp(-y_i f_t(x_i))$ using induction:
Base case: Let $t = 1$. Then (recalling $w_i^1 = \frac{1}{n}$)

$$\begin{aligned} w_i^{t+1} = w_i^2 &= w_i^1 e^{(-\alpha_1 y_i G_1(x_i))} \\ &= \frac{1}{n} e^{(-y_i f_1(x_i))} \\ &= \frac{1}{n} e^{(-y_i f_t(x_i))} \end{aligned}$$

Now assume $w_i^t = \frac{1}{n} e^{(-y_i f_{t-1}(x_i))}$. Then

$$\begin{aligned} w_i^{t+1} &= w_i^t e^{(-\alpha_t y_i G_t(x_i))} \\ &= \frac{1}{n} e^{(-y_i f_{t-1}(x_i))} e^{(-\alpha_t y_i G_t(x_i))} \\ &= \frac{1}{n} e^{(-y_i [f_{t-1}(x_i) + \alpha_t G_t(x_i)])} \\ &= \frac{1}{n} e^{(-y_i f_t(x_i))} \end{aligned}$$

6.4 $\frac{Z_{t+1}}{Z_t} = 2\sqrt{\text{err}_{t+1}(1-\text{err}_{t+1})}$

Starting from $\frac{Z_{t+1}}{Z_t}$, we find

$$\begin{aligned}
\frac{Z_{t+1}}{Z_t} &= \frac{1/n \sum_{i=1}^n \exp(-y_i f_{t+1}(x_i))}{1/n \sum_{i=1}^n \exp(-y_i f_t(x_i))} \\
&= \frac{\sum_{i=1}^n 1/n \exp(-y_i f_{t+1}(x_i))}{\sum_{i=1}^n 1/n \exp(-y_i f_t(x_i))} \\
&= \frac{\sum_{i=1}^n w_i^{t+2}}{\sum_{i=1}^n w_i^{t+1}} \\
&= \frac{\sum_{i=1}^n w_i^{t+1} \exp(-\alpha_{t+1} y_i G_t(x_i))}{\sum_{i=1}^n w_i^{t+1}}
\end{aligned}$$

Now we split the sum in the numerator into two sums: first, over i such that $y_i = G_t(x_i)$ (so $y_i G_t(x_i) = 1$), and second, over i such that $y_i \neq G_t(x_i)$ (so $y_i G_t(x_i) = -1$).

$$\begin{aligned}
\frac{Z_{t+1}}{Z_t} &= \frac{\sum_{i=1}^n w_i^{t+1} \exp(-\alpha_{t+1} y_i G_t(x_i))}{\sum_{i=1}^n w_i^{t+1}} \\
&= \frac{\sum_{i=1}^n w_i^{t+1} 1(G_t(x_i) \neq y_i) \exp(\alpha_{t+1})}{\sum_{i=1}^n w_i^{t+1}} + \frac{\sum_{i=1}^n w_i^{t+1} 1(G_t(x_i) = y_i) \exp(-\alpha_{t+1})}{\sum_{i=1}^n w_i^{t+1}} \\
&= \text{err}_{t+1} \exp(\alpha_{t+1}) + (1-\text{err}_{t+1}) \exp(-\alpha_{t+1}) \\
&= \text{err}_{t+1} \exp(1/2 \log(1/\text{err}_{t+1} - 1)) + (1-\text{err}_{t+1}) \exp(-1/2 \log(1/\text{err}_{t+1} - 1)) \\
&= \text{err}_{t+1}(1/\text{err}_{t+1} - 1)^{1/2} + (1-\text{err}_{t+1})(1/\text{err}_{t+1} - 1)^{-1/2} \\
&= (\text{err}_{t+1}^2(1/\text{err}_{t+1} - 1))^{1/2} + \text{err}_{t+1}(1/\text{err}_{t+1} - 1)(1/\text{err}_{t+1} - 1)^{-1/2} \\
&= (\text{err}_{t+1}^2(1/\text{err}_{t+1} - 1))^{1/2} + (\text{err}_{t+1}^2(1/\text{err}_{t+1} - 1))^{1/2} \\
&= 2\sqrt{(\text{err}_{t+1}(1-\text{err}_{t+1})}
\end{aligned}$$

6.5 Showing $\frac{Z_{t+1}}{Z_t} \leq \exp(-2\gamma^2)$

We show this statement is true in three steps. First, we show the function $g(a) = a(1-a)$ is monotonically increasing on [0,1/2]. Then we show that $1 - a \leq \exp(-a)$. Finally, we put it together to get the desired result.

First, consider $g(a) = a(1-a)$. Then note $g'(a) = 1 - 2a \geq 0$ for all $a \in [0, 1/2]$. Thus g is monotonically increasing over the interval [0,1/2].

Next, consider $1 - a \leq \exp(-a)$. Note the second derivative of $\exp(-a)$ is $\exp(-a) > 0$ for all a. Thus, this is a convex function. As such, it lies above its tangent at all points- in particular it lies above it's tangent at 0, so:

$$\begin{aligned}
\exp(-a) &\geq \exp(-0) + (-\exp(-0))(a-0) \\
\exp(-a) &\geq 1 + (-1)(a) \\
\exp(-a) &\geq 1 - a
\end{aligned}$$

Finally, we tackle the expression $\frac{Z_{t+1}}{Z_t}$.
First, if $\text{err}_{t+1} < 1/2 - \gamma \leq 1/2$, then (using our earlier result that $g(a)$ is monotonically increasing for $a \in [0, 1/2]$)

$$\begin{aligned}(\text{err}_{t+1})(1 - \text{err}_{t+1}) &\leq (1/2 - \gamma)(1 - (1/2 - \gamma)) \\ &\leq (1/2 - \gamma)(1/2 + \gamma)) \\ &\leq 1/4 - \gamma^2 \\ &\leq 1/4(1 - 4\gamma^2)\end{aligned}$$

Then, (by our lower bound on $\exp(-a)$) we also have

$$(\text{err}_{t+1})(1 - \text{err}_{t+1}) \leq 1/4(1 - 4\gamma^2) \leq 1/4 \exp(-4\gamma^2)$$

So substituting yields

$$\begin{aligned}\frac{Z_{t+1}}{Z_t} &= 2\sqrt{(\text{err}_{t+1})(1 - \text{err}_{t+1})} \\ &\leq 2\sqrt{1/4 \exp(-4\gamma^2)} \\ &\leq 2 \cdot 1/2\sqrt{\exp(-2\gamma^2)^2} \\ &\leq \exp(-2\gamma^2)\end{aligned}$$

6.6 Completing proof

By 2, we have $L(G, D) < Z_T$. Thus,

$$\begin{aligned}L(G, D) &< Z_T \\ L(G, D) &< \frac{Z_T}{Z_{T-1}} \cdot \frac{Z_T - 1}{Z_{T-2}} \cdots \frac{Z_2}{Z_1} \cdot \frac{Z_1}{Z_0} \cdot Z_0 \\ &\leq \exp(-2\gamma^2)^T Z_0 \\ &\leq \exp(-2T\gamma^2)\end{aligned}$$

Note, to complete this proof, we assume we start with a round 0, where $f \equiv 0$, so $Z_0 = 1$.

7. AdaBoost is FSAM with Exponential Loss

Our objective in this section is to show AdaBoost is FSAM with exponential loss.

7.1 Write first step of additive model with exponential loss

Recall the exponential loss function is $L(y, f(x)) = \exp(-y, f(y))$.
The first step in the additive model is to compute

$$(\alpha_t, G_t) = \arg\min_{\alpha, G} \sum_{i=1}^{n} L(y_i, f_{t-1}(x_i) + \alpha G(x_i))$$

Substituting in the exponential loss function yields

$$\begin{aligned}(\alpha_t, G_t) &= \arg\min_{\alpha,G} \sum_{i=1}^{n} L(y_i, f_{t-1}(x_i) + \alpha G(x_i)) \\ &= \arg\min_{\alpha,G} \sum_{i=1}^{n} \exp(-y_i(f_{t-1}(x_i) + \alpha G(x_i))) \\ &= \arg\min_{\alpha,G} \sum_{i=1}^{n} [\exp(-y_i f_{t-1}(x_i)) \exp(-y_i \alpha G(x_i)))]\end{aligned}$$

Now, from question 6, we know to set $w_i^t = 1/n \exp(-y_i f_{t-1}(x_i))$, so substitution yields

$$(\alpha_t, G_t) = \arg\min_{\alpha,G} \sum_{i=1}^{n} \left[n w_i^t \exp(-y_i \alpha G(x_i))\right] = \arg\min_{\alpha,G} \sum_{i=1}^{n} \left[w_i^t \exp(-y_i \alpha G(x_i))\right]$$

7.2 Solving for G_t for a fixed positive α

Now consider α fixed. We want to find

$$G_t = \arg\min_{G} \sum_{i=1}^{n} \left[w_i^t \exp(-y_i \alpha G(x_i))\right]$$

We proceed by splitting the sum in part 1 into two parts: first, over i such that $y_i = G(x_i)$ (so $y_i G(x_i) = 1$), and second, over i such that $y_i \neq G(x_i)$ (so $y_i G(x_i) = -1$). Then

$$G_t = \arg\min_{G} \sum_{i=1}^{n} \left[w_i^t \exp(-\alpha) 1(G(x_i) = y_i)\right] + \sum_{i=1}^{n} \left[w_i^t \exp(\alpha) 1(G(x_i) \neq y_i)\right]$$

Now consider any particular x_i: note either $G(x_i) = y_i$ or $G(x_i) \neq y_i$ (since every G in AdaBoost maps $x_i \mapsto \{1, -1\}$). In addition, note (for any $\alpha > 0$)

$$w_i^t \exp(-\alpha) < w_i^t \exp(\alpha)$$

Thus, G_t that minimizes $\sum_{i=1}^{n} \left[w_i^t 1(G(x_i) \neq y_i)\right]$ also minimizes our objective, so

$$G_t = \arg\min_{G} \sum_{i=1}^{n} \left[w_i^t 1(G(x_i) \neq y_i)\right]$$

7.3 Solving for α_t

Now we plug this G_t back into the first equation and solve for α.

$$\begin{aligned}\sum_{i=1}^{n} \left[w_i^t \exp(-y_i \alpha G(x_i))\right] &= \sum_{i=1}^{n} \left[w_i^t \exp(-\alpha) 1(G_t(x_i) = y_i)\right] + \sum_{i=1}^{n} \left[w_i^t \exp(\alpha) 1(G_t(x_i) \neq y_i)\right] \\ &= \exp(-\alpha) \sum_{i=1}^{n} \left[w_i^t 1(G_t(x_i) = y_i)\right] + \exp(\alpha) \sum_{i=1}^{n} \left[w_i^t 1(G_t(x_i) \neq y_i)\right]\end{aligned}$$

Now let's get notation under control. Let

$$c_= = \sum_{i=1}^{n} \left[w_i^t 1(G_t(x_i) = y_i) \right]$$
$$c_{\neq} = \sum_{i=1}^{n} \left[w_i^t 1(G_t(x_i) \neq y_i) \right]$$

Note, as well, that

$$c_= + c_{\neq} = \sum_{i=1}^{n} \left[w_i^t 1(G_t(x_i) = y_i) \right] + \sum_{i=1}^{n} \left[w_i^t 1(G_t(x_i) \neq y_i) \right] = \sum_{i=1}^{n} w_i^t$$

Now let's substitute:

$$\begin{aligned} \sum_{i=1}^{n} \left[w_i^t \exp(-y_i \alpha G(x_i)) \right] &= \exp(-\alpha) \sum_{i=1}^{n} \left[w_i^t 1(G_t(x_i) = y_i) \right] + \exp(\alpha) \sum_{i=1}^{n} \left[w_i^t 1(G_t(x_i) \neq y_i) \right] \\ &= \exp(-\alpha) c_= + \exp(\alpha) c_{\neq} \end{aligned}$$

To minimize with respect to α, we take the derivative and set it to 0:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\alpha} [\exp(-\alpha) c_= + \exp(\alpha) c_{\neq}] = -\exp(-\alpha) c_= + \exp(\alpha) c_{\neq} &\triangleq 0 \\ \implies \quad \exp(2\alpha) c_{\neq} &= c_= \\ \implies \quad \exp(2\alpha) &= \frac{c_=}{c_{\neq}} \\ \implies \quad \alpha &= \frac{1}{2} \log \frac{c_=}{c_{\neq}} \\ \implies \quad \alpha &= \frac{1}{2} \log \left(\frac{c_= + c_{\neq}}{c_{\neq}} - 1 \right) \\ \implies \quad \alpha &= \frac{1}{2} \log \left(\frac{\sum_{i=1}^{n} w_i^t}{\sum_{i=1}^{n} [w_i^t 1(G_t(x_i) \neq y_i)]} - 1 \right) \\ \implies \quad \alpha &= \frac{1}{2} \log \left(\frac{1}{\mathrm{err}_t} - 1 \right) \end{aligned}$$

So, we've now shown:

$$\alpha_t = \frac{1}{2} \log \left(\frac{1}{\mathrm{err}_t} - 1 \right)$$

7.4 Finding weight iterations

First, recall that $w_i^t = 1/n \exp(-y_i f_{t-1}(x_i))$.

Then note

$$\begin{aligned} w_i^{t+1} &= 1/n \exp(-y_i f_t(x_i)) \\ &= 1/n \exp(-y_i \left[f_{t-1}(x_i) + \alpha_t G_t(x_i)\right]) \\ &= 1/n \exp(-y_i f_{t-1}(x_i)) \exp(-y_i \alpha_t G_t(x_i)) \\ &= w_i^t \exp(-y_i \alpha_t G_t(x_i)) \end{aligned}$$

Thus, we've shown AdaBoost is equivalent to FSAM with exponential loss.

2 Decision Trees

2.1 Trees on the Banana Dataset

2.1.1 Visualize decision boundary

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier

# Parameters
n_classes = 2
plot_colors = "by"
plot_step = 0.02

# Load data
train = pd.read_csv('./data/banana_train.csv', header=None, names=['target', 'X'
,'Y'])
```

In [2]:

```
X = train.ix[:,1:3].as_matrix()
y = train.ix[:,0].as_matrix()
```

In [3]:

```
def relabel_y(y):
    if y == -1:
        return 0
    elif y == 1:
        return 1

v_relabel_y = np.vectorize(relabel_y)
```

In [4]:

```
# Shuffle
idx = np.arange(X.shape[0])
np.random.seed(13)
np.random.shuffle(idx)
X = X[idx]
y = y[idx]
y = v_relabel_y(y)
```

In [5]:

```
# Standardize
mean = X.mean(axis=0)
std = X.std(axis=0)
X = (X - mean) / std

# Train
clf = DecisionTreeClassifier().fit(X, y)

# Plot the decision boundary
plt.figure(figsize=(8,8))

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step), np.arange(y_min, y_max,
plot_step))

Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
cs = plt.contourf(xx, yy, Z, cmap=plt.cm.Paired)

plt.xlabel('X')
plt.ylabel('Y')
plt.axis("tight")

# Plot the training points
for i, color in zip(range(n_classes), plot_colors):
    idx = np.where(y == i)
    plt.scatter(X[idx, 0], X[idx, 1], c=color, label='Class = {}'.format(i), cma
p=plt.cm.Paired)

plt.title("Decision surface of decision tree")
plt.legend()
plt.savefig('./figures/2_1_1.png')
```

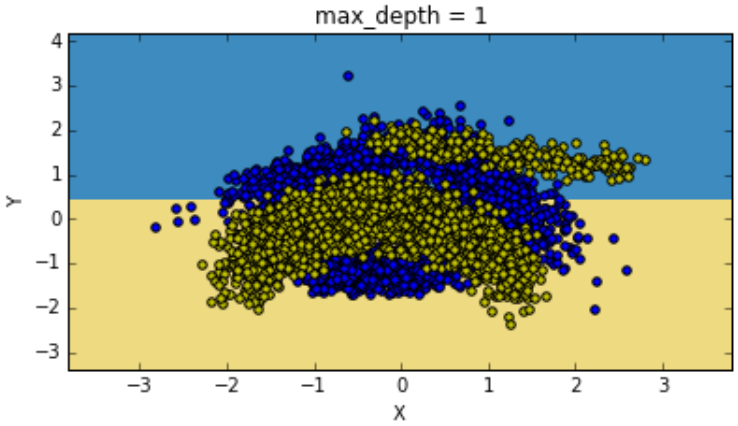
2.1.2 Comparing trees based on max_depth

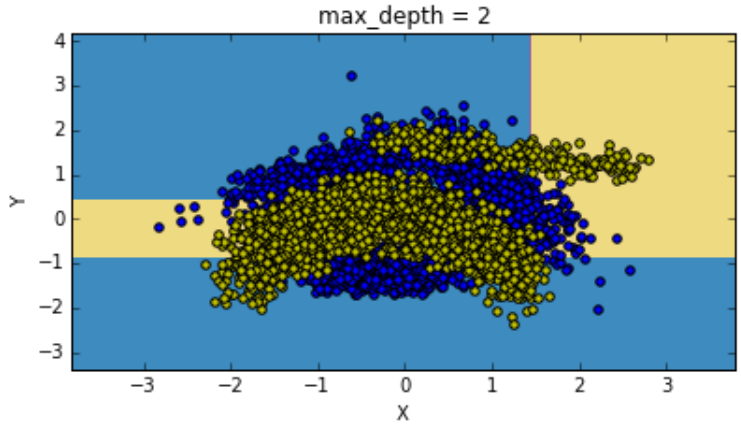
In [6]:

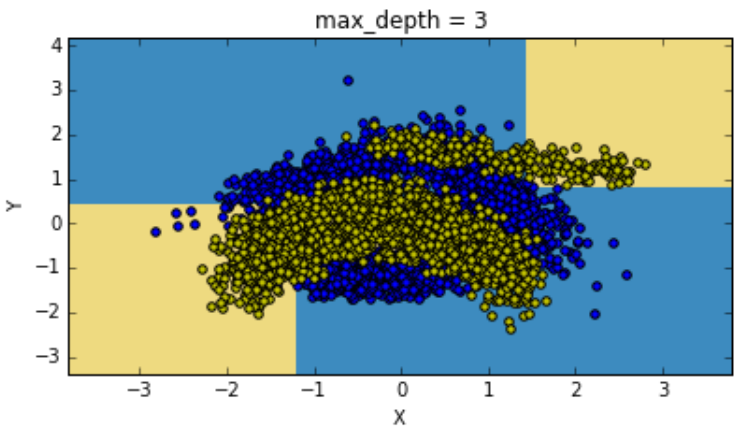
```
## Note we will also determine train/test scores for part 2.1.3
test = pd.read_csv('./data/banana_test.csv', header=None, names=['target', 'X','
Y'])
X_test = test.ix[:,1:3].as_matrix()
y_test = test.ix[:,0].as_matrix()
y_test = v_relabel_y(y_test)
mean = X_test.mean(axis=0)
std = X_test.std(axis=0)
X_test = (X_test - mean) / std
```

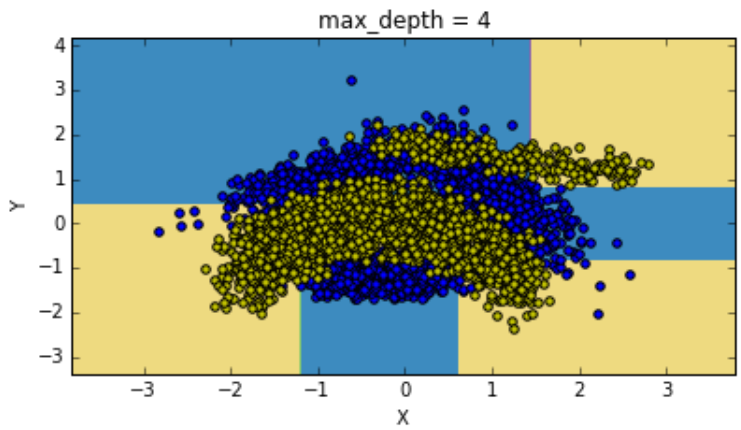
In [409]:

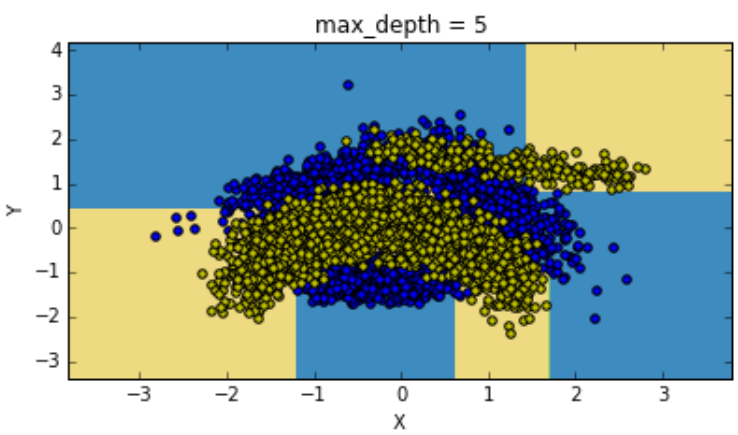
```
train_scores = []
test_scores = []

plt.figure(figsize =(14,20))
plt.subplots_adjust(hspace = .3)
for i in range(1,11):

    # Train
    clf = DecisionTreeClassifier(max_depth=i).fit(X, y)

    # Score
    train_scores.append(1.0 - clf.score(X, y))
    test_scores.append(1.0 - clf.score(X_test, y_test))

    # Plot
    plt.subplot(5,2,i)
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    cs = plt.contourf(xx, yy, Z, cmap=plt.cm.Paired)

    plt.xlabel('X')
    plt.ylabel('Y')
    plt.axis("tight")

    # Plot the training points
    for j, color in zip(range(n_classes), plot_colors):
        idx = np.where(y == j)
        plt.scatter(X[idx, 0], X[idx, 1], c=color, label='Class = {}'.format(j),
cmap=plt.cm.Paired)

    plt.title("max_depth = {}".format(i))
plt.savefig('./figures/2_1_2.png')
```

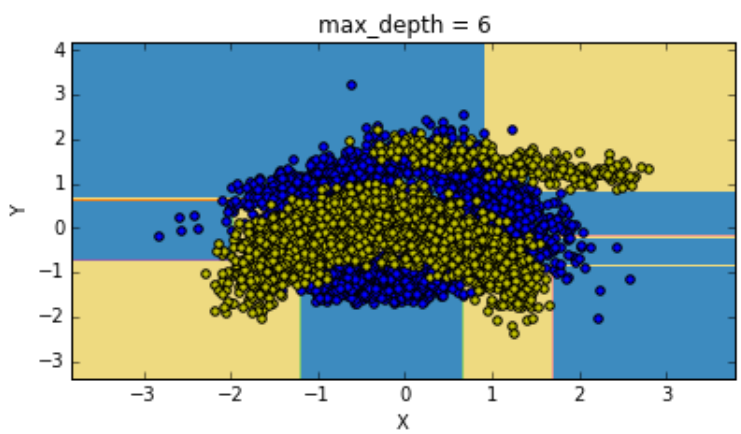


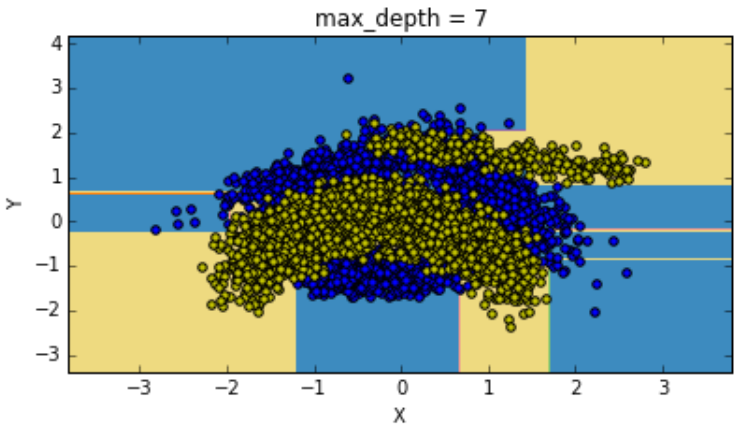


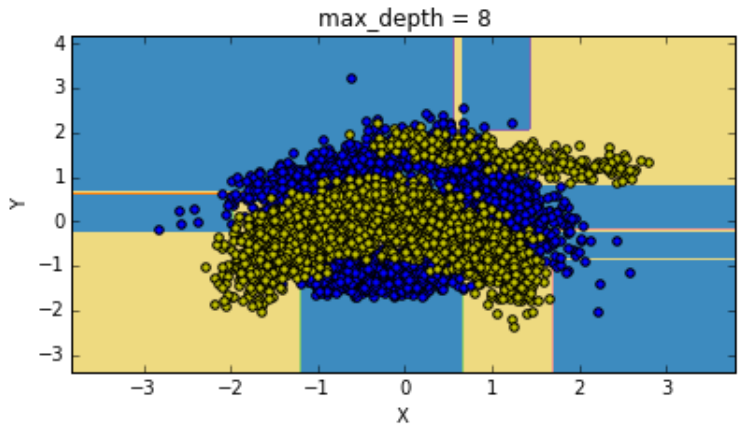


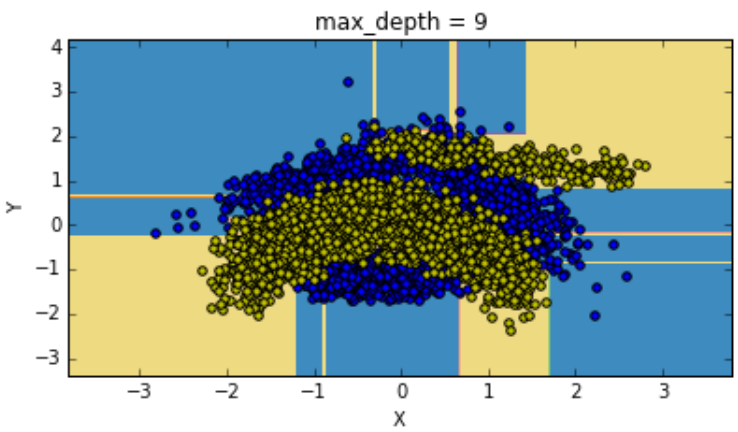


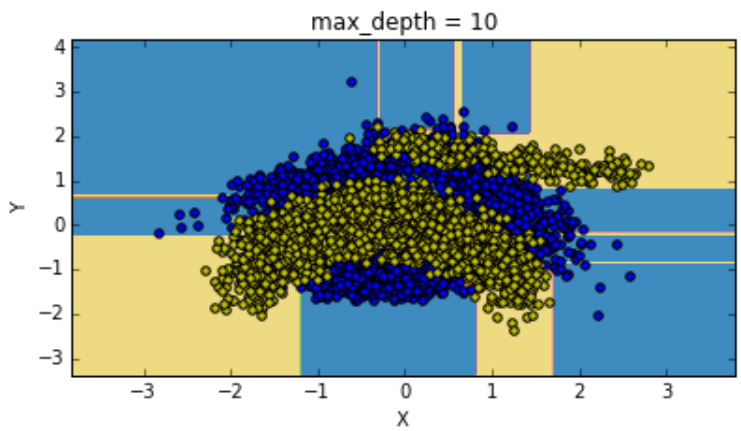








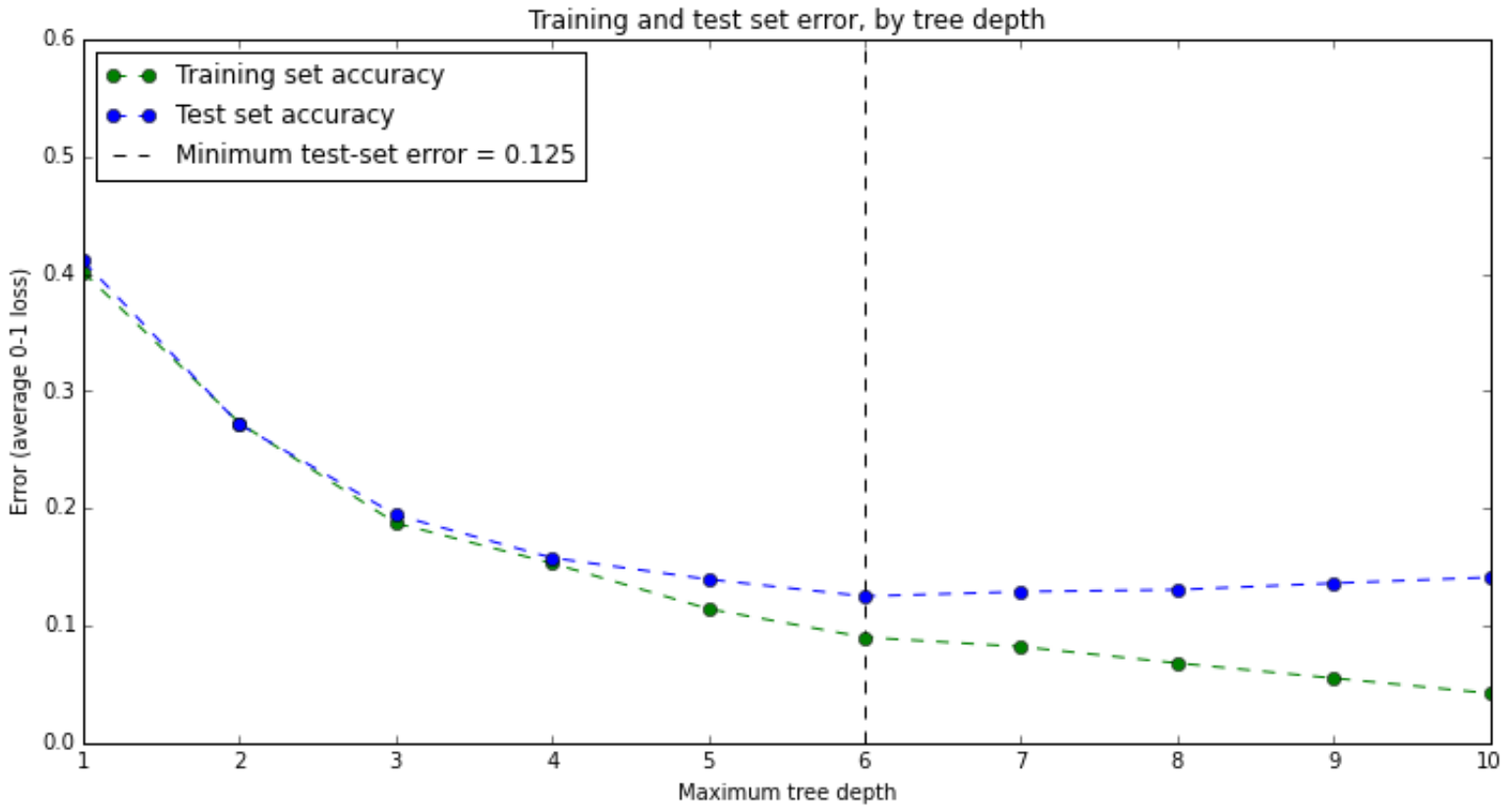




2.1.3 Training and test errors vs. max_depth

In [410]:

```
plt.figure(figsize=(12,6))
plt.plot(range(1,11),train_scores, color='green', linestyle='dashed', marker='o'
, label='Training set accuracy')
plt.plot(range(1,11),test_scores, color='blue', linestyle='dashed', marker='o',
label='Test set accuracy')
plt.vlines(test_scores.index(min(test_scores))+1, 0, 0.6, colors='k', linestyles
='dashed', label='Minimum test-set error = {}'.format(min(test_scores)))
plt.legend(loc = 'upper left')
plt.title('Training and test set error, by tree depth')
plt.ylabel('Error (average 0-1 loss)')
plt.xlabel('Maximum tree depth')
plt.savefig('./figures/2_1_3.png')
```



2.1.4 Optimizing other hyperparameters

In [7]:

```
from sklearn import grid_search

param_grid = [
  {'criterion': ['gini', 'entropy'],
   'max_depth' : range(1,11),
   'min_samples_split' : np.logspace(1,9,9, base=2),
   'min_samples_leaf' : np.logspace(0,9,10, base=2)}]

dt = DecisionTreeClassifier()
clf = grid_search.GridSearchCV(dt, param_grid)
clf.fit(X, y)
```

Out[7]:

```
GridSearchCV(cv=None, error_score='raise',
       estimator=DecisionTreeClassifier(class_weight=None, criterion
='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf
=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            random_state=None, splitter='best'),
       fit_params={}, iid=True, loss_func=None, n_jobs=1,
       param_grid=[{'min_samples_split': array([   2.,    4.,    8.,
16.,   32.,   64.,  128.,  256.,  512.]), 'criterion': ['gini', 'ent
ropy'], 'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'min_samples_l
eaf': array([   1.,    2.,    4.,    8.,   16.,   32.,   64.,  128.,
256.,  512.])}],
       pre_dispatch='2*n_jobs', refit=True, score_func=None, scoring
=None,
       verbose=0)
```

In [8]:

```
clf.best_score_
```

Out[8]:

```
0.89114285714285713
```

In [9]:

```
clf.best_params_
```

Out[9]:

```
{'criterion': 'gini',
 'max_depth': 9,
 'min_samples_leaf': 8.0,
 'min_samples_split': 2.0}
```

In [12]:

```
1 - clf.score(X_test, y_test)
```

Out[12]:

```
0.13722222222222225
```

3 AdaBoost

3.1 Implementation

3.1.1 Implementing with depth 3 trees as base classifier

In [415]:

```
##Need -1, 1 y's for Ada_boost:
def inv_relabel_y(y):
    if y == 1:
        return 1
    elif y == 0:
        return -1
    else:
        return y

v_inv_relabel_y = np.vectorize(inv_relabel_y)
```

In [416]:

```
y = v_inv_relabel_y(y)
y_test = v_inv_relabel_y(y_test)
```

In [417]:

```
##Mutates input weights
def update_weights(weights, alpha, loss_vector):
    weights[loss_vector == 1,] = weights[loss_vector == 1,]*np.e**(alpha)
    return weights
```

In [418]:

```
def AdaBoost(X_train, y_train, num_iter):
    ## Initialize weights
    n = X_train.shape[0]
    weights = np.ones(n)/float(n)

    ## Initialize lists to store alphas and trees
    alphas = []
    trees = []

    for m in range(num_iter):
        clf = DecisionTreeClassifier(max_depth=3).fit(X_train, y_train, sample_w
eight=weights)
        loss_vector = np.not_equal(clf.predict(X_train),y_train)
        err = 1.0/(np.sum(weights))*np.dot(weights,loss_vector)
        alpha = np.log((1.0-err)/err)
        update_weights(weights, alpha, loss_vector)
        alphas.append(alpha)
        trees.append(clf)

    return weights, alphas, trees
```

In [419]:

```
def predict_label_from_classifiers(X_s, alphas, trees):
    predictions = np.zeros(X_s.shape[0])
    for m in range(len(alphas)):
        predictions = predictions + alphas[m]*trees[m].predict(X_s)

    return np.sign(predictions)
```

In [420]:

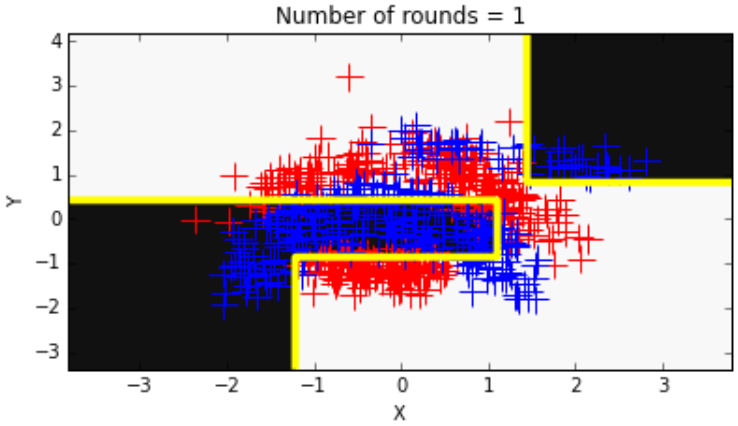
```
def unsigned_score(X_s, alphas, trees):
    predictions = np.zeros(X_s.shape[0])
    for m in range(len(alphas)):
        predictions = predictions + alphas[m]*trees[m].predict(X_s)

    return predictions
```

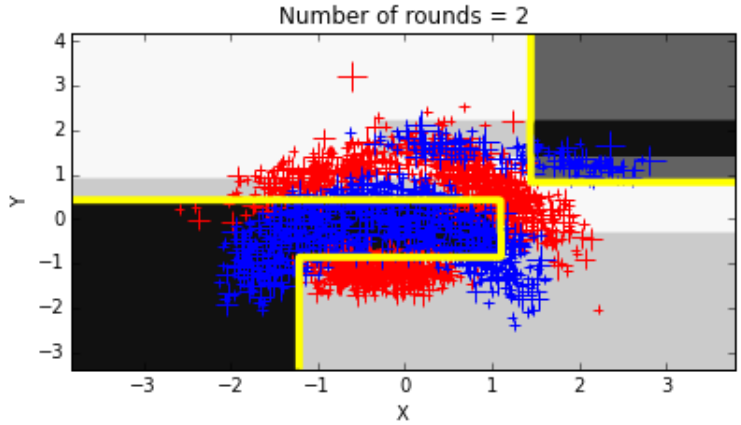
In [421]:

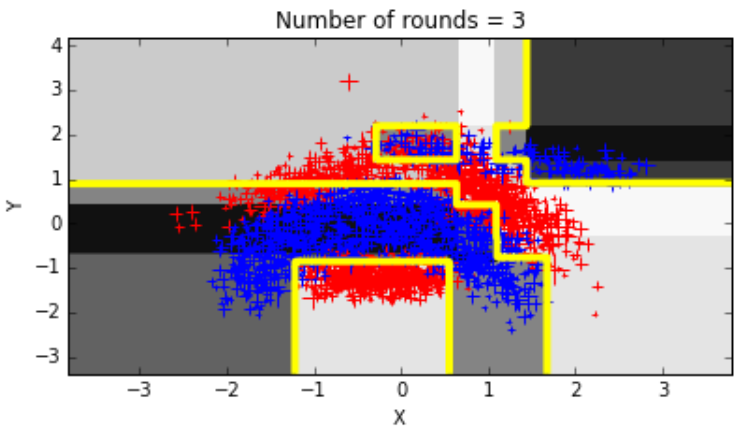
```
def prediction_error(prediction, truth):
    return np.sum(np.not_equal(prediction, truth))/float(len(truth))
```

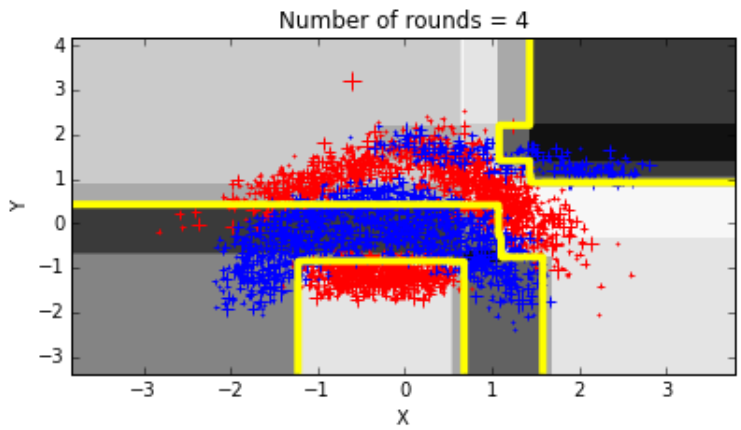
In [422]:

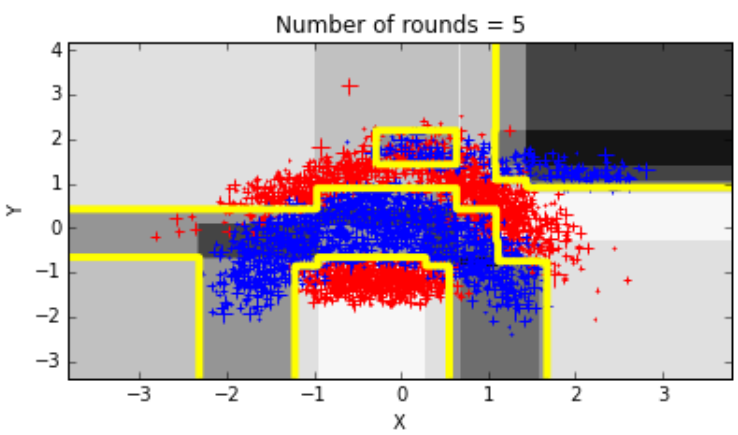
```
train_errors = []
test_errors = []

plt.figure(figsize =(14,20))
plt.subplots_adjust(hspace = .3)

for num_iter in range(1,11):

    # Train
    weights, alphas, trees = AdaBoost(X, y, num_iter)

    # Scale the weights for plotting
    weights = (weights-np.min(weights))/np.max(weights) * 300

    #Score
    y_hat_train = predict_label_from_classifiers(X, alphas, trees)
    train_errors.append(prediction_error(y_hat_train, y))
    y_hat_test = predict_label_from_classifiers(X_test, alphas, trees)
    test_errors.append(prediction_error(y_hat_test,y_test))

    plt.subplot(5,2,num_iter)
    Z = unsigned_score(np.c_[xx.ravel(), yy.ravel()], alphas, trees)
    Z = Z.reshape(xx.shape)
    cs = plt.contourf(xx, yy, Z, cmap=plt.cm.Greys)
    cs_lines = plt.contour(xx, yy, Z, levels=[0.0],linewidths=4, colors='#ffff00
')

    plt.xlabel('X')
    plt.ylabel('Y')
    plt.axis("tight")

    # Plot the training points
    for j, color in zip([-1,1], 'rb'):
        idx = np.where(y == j)
        plt.scatter(X[idx, 0], X[idx, 1], c=color, marker='+', s=weights)

    plt.title("Number of rounds = {}".format(num_iter))

plt.savefig('./figures/3_1_2.png')
```

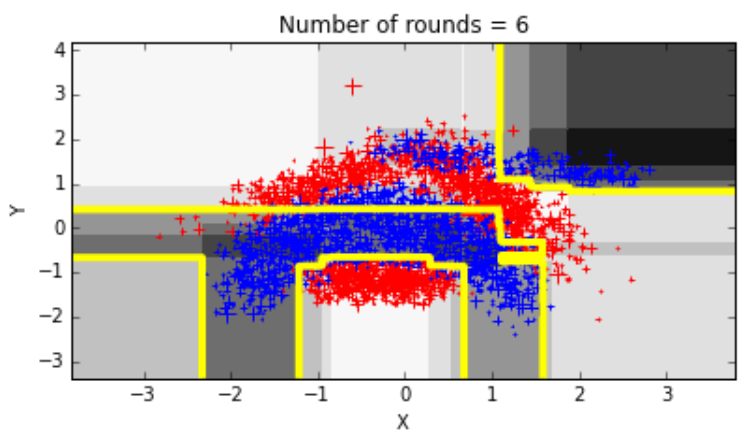

Number of rounds = 1
X
Y

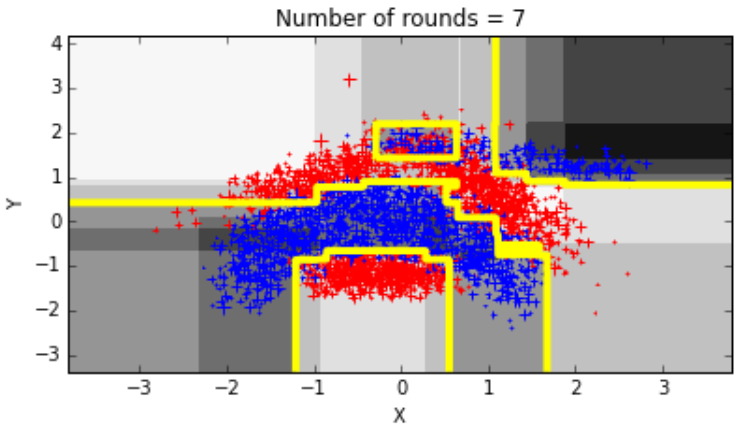

Number of rounds = 2
X
Y

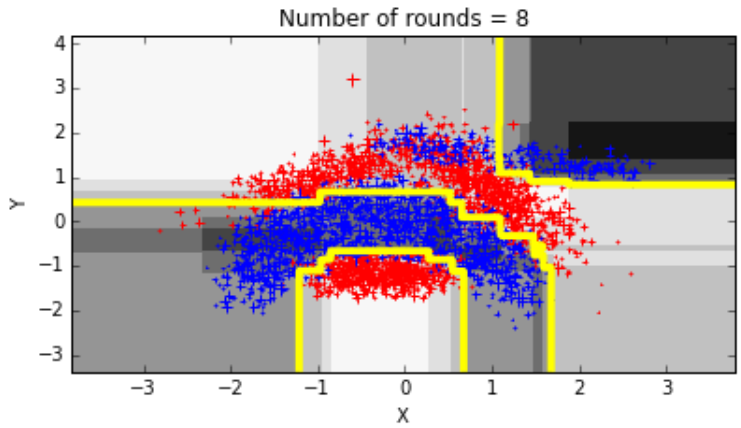

Number of rounds = 3
X
Y

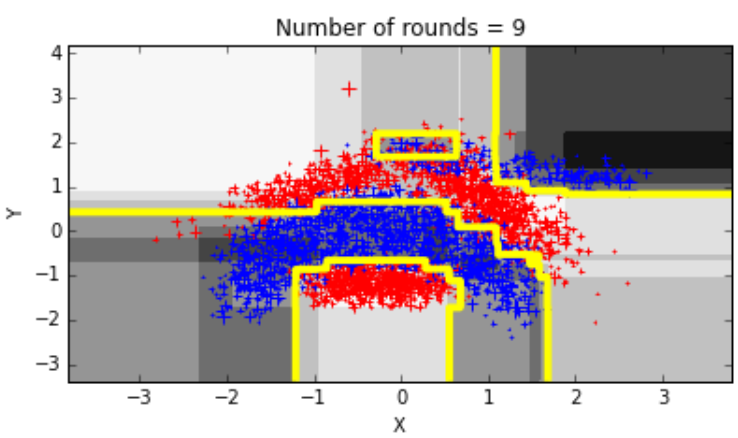

Number of rounds = 4
X
Y

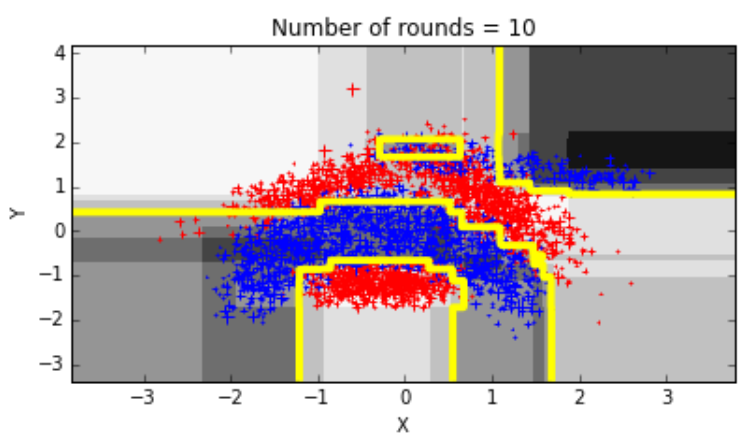

Number of rounds = 5
X
Y


Number of rounds = 6
X
Y


Number of rounds = 7
X
Y


Number of rounds = 8
X
Y


Number of rounds = 9
X
Y


Number of rounds = 10
X
Y

In [423]:

```
plt.figure(figsize=(12,6))
plt.plot(range(1,11),train_errors, color='green', linestyle='dashed', marker='o'
, label='Training set accuracy')
plt.plot(range(1,11),test_errors, color='blue', linestyle='dashed', marker='o',
label='Test set accuracy')
plt.vlines(test_errors.index(min(test_errors))+1, 0, 0.3, colors='k', linestyles
='dashed', label='Minimum test-set error = {}'.format(round(min(test_errors),3))
)
plt.legend(loc = 'upper left')
plt.title('Training and test set accuracy, by number of rounds of AdaBoost')
plt.ylabel('Accuracy (1 - average 0-1 loss)')
plt.xlabel('Number of rounds of AdaBoost')
plt.savefig('./figures/3_1_3.png')
```