Problem 1

Setup: Suppose that you have a black box learning algorithm $A$ for optimizing zero-one loss: For any distribution $D'$ over $X \times \{0, 1\}$, $A$ takes a set of training examples from $D'$, and produces a classifier $f : X \to \{0, 1\}$ optimized for $E_{(x, y) \sim D'} 1[f(x) \neq y]$. (Here $1[\cdot]$ is the indicator function, which is 1 when its argument is true, and 0 otherwise.)

You have a distribution $D$ over $X \times \{0, 1\}$, but the loss function you care about is asymmetric. You want to learn a classifier $h : X \to \{0, 1\}$ minimizing

$$\rho(h) = E_{(x, y) \sim D} \{10^{y-1} \cdot 1[h(x) \neq y]\}.$$  

In other words, you care about false negatives (predicting 0 when the true label is 1) 10 times more than about false positives (predicting 1 when the true label is 0).\footnote{Think about predicting the presence ($y = 1$) or absence ($y = 0$) of a disease based on lab results $x$. It may be 10 times better to have a false alarm rather than let the disease go unnoticed.}

Problem: How do you use $A$ to optimize $\rho$ on $D$? You can’t modify $A$ (you don’t have the source code or the code is too complicated to be tweaked).

Hints: You can tweak the training set sampled from $D$ before feeding it into $A$, essentially re-wighting $D$ so that minimizing the symmetric rate of errors on the re-weighted distribution $D'$ is equivalent to minimizing $\rho$ on $D$. (What’s the optimal $D'$?)

At training time, you need to convert a sample $S$ from $D$ into a sample $S'$ from $D'$. Given $S'$, the black box $A$ returns a classifier $f$ minimizing the symmetric rate of errors on $D'$. At test time, you can use predictions made by $f$ (on any examples of your choice) to construct your prediction on a test example drawn from $D$. Depending on your solution, you can simply output $f$’s output on the test example.

You are allowed to train multiple classifiers using $A$ (by feeding $A$ different training sets). You can use these classifiers in an arbitrary way at test time. The only thing you are not allowed to do is to tweak $A$ itself.
Solution: Let $c(y)$ be the cost of misclassifying any example with label $y$. In our problem, $c(y) = 10^{y-1}$. The solution should be guided by the following observation: For any distribution $D$ over $X \times \{0,1\}$ and any $w \geq 1$, we can define
\[
D'(x, y) = \frac{c(y)}{W} D(x, y),
\]
where $W = E_{(x,y) \sim D} c(y)$ is just the expected misprediction cost of a random example from $D$, so that for all classifiers $f : X \rightarrow \{0,1\}$
\[
E_{(x,y) \sim D} 1[f(x) \neq y] = \frac{1}{W} E_{(x,y) \in D} [c(y) \cdot 1[f(x) \neq y]].
\]
To see that the observation is true, simply observe that
\[
E_{(x,y) \sim D} [c(y) \cdot 1[f(x) \neq y]] = \sum_{(x,y) \in X \times \{0,1\}} D(x,y) \cdot c(y) \cdot 1[f(x) \neq y]
= W \sum_{(x,y) \in X \times \{0,1\}} D'(x,y) \cdot 1[f(x) \neq y]
= W E_{(x,y) \sim D'} 1[f(x) \neq y],
\]
assuming that $X$ is finite. Now, the natural thing to do is to reweight the distribution $D$ in our training set according to the weights, to produce a sample from $D'$. Several simple sampling schemes are reasonable. An acceptable solution is to define a probability distribution over the training set according to the weights, to produce a sample from $D'$.

Problem 2

Setup: Let $X = \{0,1\}^n$ be the set of all $n$ bit input strings, and let $Y = \{0,1\}$. Consider a distribution $D$ over $X \times Y$ specified by $D(x,y) = D(x)D(y \mid x)$: The marginal distribution over $X$ is uniform. Thus for every $x \in X$, $D(x) = 2^{-n}$. For every $x \in X$, the conditional distribution over $Y$ given $x$ puts all its probability mass on $y' = \{(x_{n-1} + x_n) \mod 2\}$ (parity of the first two bits of $x$); i.e., $D(y' \mid x) = 1$ and $D(1-y' \mid x) = 0$. Thus the conditional probability distribution is independent of the first $n-2$ bits of $x$.

Problem: You get a set of $N$ independent examples from $D$ and you are using a decision tree learning algorithm $A$ to produce a classifier $f : X \rightarrow \{0,1\}$. The algorithm $A$ is generic, i.e., it does not know $D$. To be specific,

- all test are of the form “Is $x_i = 1$?” for $i \in \{1, \ldots, n\}$,
- information gain criteria is used to select tests, with ties broken randomly,
- there is no lookahead,
- the tree can be post-pruned after it has been grown to zero error.

What is the smallest expected number of examples $N$ (big-O precision is fine here) such that the learned tree has expected error rate 0 on $D$ (where the expectation is with respect to the draw of the training set $S_N$ of size $N$ and the randomness in the algorithm)? Explain your answer. What is the expected entropy of the class label in $S_N$ (the expectation is with respect to the draw of $S_N$)?

Solution: The expected number of examples with label 0 in $S_N$ is $N/2$, thus the expected original entropy about the class is 1. In expectation over the draw of $S_N$, all $n$ tests have zero information gain, so a random $x_i$ will be chosen as the root. The expected information gain of any test will remain 0 until one of $x_{n-1}$ or
$x_n$ is chosen as a test on each decision path. After that, the expected information gain of the other test will be (effectively) 1, so splitting on it will result in (effectively) 0 error.\footnote{There is, of course, a slim chance that we get a non-representative sample from $D$; for example, we may get the same example $N$ times. This is not the point of this exercise. Precision is not always the same as accuracy.} Thus the number of samples $N$ needs to be large enough so that the tree can’t grow to zero training error without testing both $x_{n-1}$ and $x_n$ on every decision path. Consequently, with high probability over the draw of the training set, the expected $N$ must be on the order of $O(2^n)$. Such level of precision is sufficient.

Problem 3

Show that for any $D$ over $X \times \mathbb{R}$ and any $x \in X$,

$$\arg\min_{y'} E_{y \sim D| x}(y - y')^2 = E_{y \sim D| x}[y].$$

Here $D | x$ is the conditional distribution over $\mathbb{R}$ given $x$. (Square brackets are simply delimiters; they don’t have semantic content here.)

**Proof:** To simplify notation, let $P$ denote $D | x$. We have

$$E_{y \sim P}(y - y')^2 = E_{y \sim P}[y^2] - 2y'E_{y \sim P}[y] + (y')^2.$$

Now the first term $E_{y \sim P}[y^2]$ is the same for all $y'$, so it doesn’t affect the argmix. We want $\arg\min_{y'} (y')^2 - 2y'E_{y \sim P}[y]$. Taking the derivative with respect to $y'$ and setting it to zero gives $y' = E_{y \sim P}[y]$, completing the proof. \qed