Problem 3.41

(a) The low-degree algorithm is this: $\forall S \subseteq [n]$ such that $|S| < k$, use $\mathrm{FOURIER(S)}$ to estimate $\hat{f}(S)$ up to accuracy ϵ with confidence δ and then output $\mathrm{sign}\left(\sum_{S:|S|<k} \widetilde{\hat{f}(S)} X_S(x)\right)$. $\mathrm{FOURIER(S)}$ samples $O\left(\frac{1}{\epsilon_1^2} \cdot \log\left(\frac{1}{\delta_1}\right)\right)$ each time, where $\epsilon_1^2 = \sqrt{\frac{\epsilon}{2 \cdot \binom{n}{<k}}}$ and $\delta_1 = \frac{1}{3} \cdot \frac{1}{\binom{n}{<k}}$. So, $O\left(\frac{1}{\epsilon_1^2} \cdot \log\left(\frac{1}{\delta_1}\right)\right) = O\left(\frac{2n^k}{\epsilon} \cdot 3n^k\right) = poly\left(n^k, \frac{1}{\epsilon}\right)$. We can just use the same $poly\left(n^k, \frac{1}{\epsilon}\right)$ examples each time (call it the batch $\mathfrak{E}$).

(b) The final hypothesis is

$$h(y) = \mathrm{sign}\left(\sum_{S:|S|<k} \widetilde{\hat{f}(S)} X_S(y)\right) = \mathrm{sign}\left(\sum_{S:|S|<k} \left(\frac{1}{|\mathfrak{E}|} \sum_{(x,f(x)) \in \mathfrak{E}}^{r} f(x) X_S(x)\right) X_S(y)\right)$$

where

$$r = poly\left(n^k, \frac{1}{\epsilon}\right)$$

Because $\frac{1}{r}$ is always positive, it has no effect on the sign of the expression and thus can be eliminated.

$$h(y) = \mathrm{sign}\left(\sum_{S:|S|<k} \left(\sum_{(x,f(x)) \in \mathfrak{E}}^{r} f(x) X_S(x)\right) X_S(y)\right)$$
$$= \mathrm{sign}\left(\sum_{(x,f(x)) \in \mathfrak{E}} f(x) \sum_{S \subseteq [n]\ s.t. |S|<k} X_S(x) X_S(y)\right)$$

I don't really see how each example's weight only depends on its Hamming distance from y because it also depends on k, as shown in this expression. It does *include* the Hamming distance, however, because when $|S| = 1$, $\sum_{S \subseteq [n]\ s.t. |S|=1} X_S(x) X_S(y) = n - 2h$, where h is the Hamming distance between x and y.

Problem 4.4

(a) A DNF of width $\log(s)$ can be computed by a depth $\log(s)$ decision tree. Using Exercise 3.30, it becomes $\hat{\|}f\hat{\|}_\infty \geq \frac{1}{2^{\log(s)}} = \frac{1}{s}$, as $|b| = 1$ because our function can only output either -1 or 1. So, $\hat{\|}f\hat{\|}_\infty$ is $\Omega(1/s)$. Then using proposition 4.9, we get that our function is computable by a DNF of width $\log(s) - \log(\epsilon)$ due to the law of logs, which then is $\log(s) + O(1)$ because $\log(\epsilon)$ is a constant. We prove the claim by putting these two together using the fact that $|\hat{f}(S)| \geq \hat{\|}f\hat{\|}_\infty$.

(b) We learned in class that if function f is computable by a size s DNF, then f is ϵ-concentrated to degree $k = O\left(\log\left(\frac{s}{\epsilon}\right) \cdot \log\left(\frac{1}{\epsilon}\right)\right)$. In this case, $\epsilon = \frac{1}{2} - \Omega\left(\frac{1}{s}\right)$. The Low-Degree algorithm says for all f in a concept class F such that f is $\frac{\epsilon}{2}$ concentrated up to degree k, F can be learned in time $poly(n^k, 1/\epsilon)$ with error ϵ. In our case, $n^k = n^{\log\left(\frac{s}{\frac{1}{2}-\Omega\left(\frac{1}{s}\right)}\right)\cdot\log\left(\frac{1}{\frac{1}{2}-\Omega\left(\frac{1}{s}\right)}\right)}$. We know that by big Omega notation, $\frac{1}{2} - \Omega\left(\frac{1}{s}\right) < \frac{1}{2}$, and s is constant for a fixed function f, so n^k becomes $\mathrm{poly}(n)$. Similarly, $\frac{1}{\epsilon}$ is $\frac{1}{\frac{1}{2}-\Omega\left(\frac{1}{s}\right)}$, which becomes $\mathrm{poly}(s)$. Together, F is learnable in time $\mathrm{poly}(n, s)$ with error $\frac{1}{2} - \Omega\left(\frac{1}{s}\right)$.

Problem 4.19

(a) By "doesn't falsify T_i", I'm taking this to mean "making sure T_i is not constantly false". We already know $(T_i)_{J|z}$ is not constant, which means there's no false restricted variables in T_i because otherwise, T_i will just always be false, as it's a conjunction. To make sure $(T_i)_{R'}$ is not always false, all you have to do is, if it's x_j in T_i , set $x_j = \mathrm{true}$; if it's $\overline{x_j}$, set $x_j = \mathrm{false}$.

(b) For a restriction to be bad, at least one T_i has to be neither constantly true nor constantly false, and for that T_i to be "bad" (i.e. neither constantly true or false), at least one variable in it has to be free. A restriction R' could be arrived at from different initial Rs if the Rs' j-th bit as defined by part (a) is a restricted bit in R' and everything else matches R'. R' can be arrived at from a maximum of w original restrictions because the most literals a clause can have is w. This situation happens like this, where c is a restricted variable:

$$(x_1 \wedge c \wedge c \wedge c) \vee (\dots) \vee \dots$$
$$(c \wedge \overline{x_2} \wedge c \wedge c) \vee (\dots) \vee \dots$$
$$(c \wedge c \wedge x_3 \wedge c) \vee (\dots) \vee \dots$$

(c)

$$\mathbb{P}[(\boldsymbol{J}|\boldsymbol{z}) = R] = \delta^{|J|} \cdot \left(\frac{1-\delta}{2}\right)^{n-|J|}$$

$$\mathbb{P}[(\boldsymbol{J}|\boldsymbol{z}) = R'] = \delta^{|J|-1} \cdot \left(\frac{1-\delta}{2}\right)^{n-|J|+1}$$

$$\frac{\mathbb{P}[(\boldsymbol{J}|\boldsymbol{z}) = R]}{\mathbb{P}[(\boldsymbol{J}|\boldsymbol{z}) = R']} = \frac{\delta^{|J|} \cdot \left(\frac{1-\delta}{2}\right)^{n-|J|}}{\delta^{|J|-1} \cdot \left(\frac{1-\delta}{2}\right)^{n-|J|+1}} = \frac{2\delta}{1-\delta}$$

(d) For $\delta \in [0, \frac{1}{3}]$, $\frac{2\delta}{1-\delta} \leq 3\delta$. And since you can arrive at a R' from at most w bad restrictions, $\mathbb{P}[(\boldsymbol{J}|\boldsymbol{z}) \text{ is bad}] \leq 3\delta w$.