# Set 2

Claire Goeckner-Wald

October 10, 2016

## Hoeffding Inequality

**1)** B. 0.01

See the Hoeffding code. For 100,000 repititions, the average value of $v_{min}$ was about 0.0376. This makes sense because we are each time choosing the coin with the minimum number of heads, skewing the distribution.

**2)** D. $c_1$ and $c_{rand}$

See the Hoeffding code. For 100,000 repititions, the average value of $c_1$ and $c_{rand}$ were about 0.5. They satisify the single-bin Hoeffding Inequality because they closely resemble the $\mu$ value of 0.5. Because these are fair coins, $\mu$, the true distribution of the number of heads in 10 coin flips, should be approximately 0.5. The values $v_1$, $v_{min}$ and $v_{rand}$ are different ways of sampling the (coin distribution) bin. But, only $c_1$ and $c_{rand}$ approximate $\mu$.

## Error and Noise

**3)** E. $\lambda\mu + (1-\lambda)(1-\mu)$

Function $h$ approximates $f$ with error $\mu$. But there is a $1-\lambda$ chance that $f$ is wrong. We can determine the probability that $h$ incorrectly approximates $y$ with a decision tree. Note that the functions are binary. This means that if $h$ incorrectly approximates a corrupted $f$, then $h$ has correctly found $y$.

**4)** B. 0.5

$$\begin{aligned} P(h(x) \neq y) &= \lambda\mu + (1-\lambda)(1-\mu) \\ &= \lambda\mu + (1-\lambda-\mu+\lambda\mu) \\ &= 2\lambda\mu - \lambda - \mu + 1 \\ &= \mu(2\lambda - 1) - \lambda + 1 \\ 2\lambda - 1 &= 0 \\ 2\lambda &= 1 \\ \lambda &= \frac{1}{2} \end{aligned}$$

Thus, when $\lambda = \frac{1}{2}$, $\mu$ cannot affect the output of $P(h(x) \neq y)$.

## Linear Regression

**5)** C. 0.01

See the linear regression code. For 10,000 repititions, the average in-sample error was 0.039185.

**6)** C. 0.01

See the linear regression code. For 10,000 repititions, the average out-sample error was 0.0484795.

**7)** A. 1

See the linear regression code. For 1,000 repititions, the average numbers of modified PLA iterations required is 5.334

## Nonlinear Transformation

**8)** D. 0.5

See the non-linear transformation code. For 1,000 repititions, the average in-sample error for the non-transformed data was 0.51429. This makes sense because the data is not yet linearly separable.

**9)** A. $\text{sign}(-1 - 0.05x_1 + 0.08x_2 + 0.13x_1x_2 + 1.5x_1^2 + 1.5x_2^2)$

See the non-linear transformation code. For 1000 test data points, the functions a - e had accuracy rates of 0.952, 0.693, 0.677, 0.614, and 0.547, respectively. Thus, [a] most resembles the hypothesis found. This makes sense because the algorithm should weight the $x_1^2$ and $x_2^2$ strongly. But, since there's a .1 chance of classification corruption, it's not perfectly weighted.

**10)** A. 0

See the non-linear transformation code. For 1,000 repititions, the average out-sample error for the transformed data was 0.033848. This makes sense because the data has been transformed so that the linear regression algorithm can sensibly work with it. That is, the data now contains the data points $x_1^2$ and $x_2^2$.