Suggested reading:

- *Elements of Statistical Learning* (by Hastie, Tibshirani, and Friedman): Chapter 1 (pages 1-8).
- *Learning from Data* (by Abu-Mostafa, Magdon-Ismail, Lin): Chapter 1. This covers much of the same ground we did in our "First look at generalization", and does so in a very conversational manner.
- "Introduction to Statistical Learning Thoery" by Bousquet, Boucheron, and Lugosi: Sections 1-3 (first 13 pages). Ignore Section 2.1 for now. The beginning of this paper provides an overview of using concentration bounds to bound the performance of empirical risk minimization in the context of a finite set of hypotheses. You can download the paper on the course web page.

Problems:

1. Suppose that we have some number m of coins. Each coin has the same probability of landing on heads, denoted p. Suppose that we pick up each of the m coins in turn and for each coin do n independent coin tosses. Note that the probability of obtaining exactly k heads out of n tosses for any given coin is given by the binomial distribution:

$$\mathbb{P}[k|n,p] = \binom{n}{k} p^k (1-p)^{n-k}.$$

For each series of coin tosses, we will record the results via the empirical estimates given by

$$\widehat{p}_i = \frac{\text{number of times coin } i \text{ lands on heads}}{n}.$$

   (a) Assume that $n = 10$. If all the coins have $p = 0.05$, compute a formula for the exact probability that *at least* one coin will have $\widehat{p}_i = 0$. (This may be easier to calculate by instead computing the probability that this *does not* occur.) Give a table containing the values of this probability for the cases of $m = 1$, $m = 1,000$, and $m = 1,000,000$. Repeat for $p = 0.75$.

   (b) Now assume that $n = 10$, $m = 2$, and that $p = 0.5$ for both coins. Compute (exactly, via a formula) and then plot/sketch

$$\mathbb{P}\left[\max_i |\widehat{p}_i - p| > \epsilon\right]$$

as a function of $\epsilon \in [0,1]$. (Note that if $n = 10$, $\widehat{p}_i$ can take only 11 discrete values, so your plot will have discrete jumps at certain values as ϵ changes.) On the same plot, show the bound that results from applying the Hoeffding inequality together with the union bound.

2. (a) Suppose that X is a Gaussian random variable with zero mean and variance σ^2:

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}.$$

Find a tail bound for X using the Chernoff bounding method (see page 9 of the notes on concentration inequalities). In other words, fill in the right hand side below with an expression that depends on t (and σ^2):

$$\mathbb{P}[X > t] \;\leq\; ???.$$

To make this bound as good as possible, optimize over your choice of λ. Expressions for the moment generating function of a Gaussian random variable are easy to come by (e.g., in the "Normal distribution" entry in Wikipedia).

(b) Suppose that $X_1, X_2, \ldots, X_m$ are iid Gaussian random variables with mean 0 and variance σ^2. Using your answer for part (a) and the union bound, find a bound for

$$\mathbb{P}\left[\max_{i=1,\ldots,m} X_i > t\right] \;\leq\; ???.$$

(c) For X_i as in part (b), complete the following sentence: With probability at least 0.9,

$$\max_{i=1,\ldots,m} X_i \;\leq\; ???.$$

(d) Using Python, create histograms for the random variable

$$Z = \max_{i=1,\ldots,m} X_i, \quad X_i \sim \text{Normal}(0,1),$$

for $m = 10^\beta$ for $\beta = 3, 4, 5, 6$. The code in `hist-example.py` should help you get started. Discuss in the context of your answer to part (c). Turn in plots of your histograms along with your comments.

3. Consider a binary classification problem involving a single (scalar) feature x and suppose that $X|Y = 0$ and $X|Y = 1$ are continuous random variables with densities given by

$$f_{X|Y}(x|0) = \begin{cases} e^{-x} & x \geq 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$f_{X|Y}(x|1) = \begin{cases} 1 & x \in [a, a+1] \\ 0 & \text{otherwise,} \end{cases}$$

respectively, where $a \geq 0$ is given. Furthermore, suppose that $\mathbb{P}[Y = 0] = \mathbb{P}[Y = 1] = \frac{1}{2}$.

(a) Sketch $f_{X|Y}(x|0)$ and $f_{X|Y}(x|1)$.

(b) Derive and state the optimal classification rule (in terms of minimizing the probability of error/risk).

(c) Calculate the Bayes risk for this classification problem in terms of the parameter a.

(d) Suppose that we do not know the true value of parameter a but have an estimate $\widehat{a}$. Determine the risk for the classifier that results from replacing a with $\widehat{a}$ in the classification rule you derived in part (b), i.e., assume that the data is still generated according to the distributions $g_0(x)$ and $g_1(x)$, but we use the classification rule based on $\widehat{a}$. (Your answer should be in terms of a and the estimate $\widehat{a}$, and you may assume that $\widehat{a} \geq 0$.)

(e) Suppose that we are given a guarantee that $\mathbb{P}[|a-\widehat{a}| \geq \epsilon] \leq \delta$ for some $\epsilon \geq 0$ and $\delta \in (0,1)$. Give an upper bound on worst-case possible risk in terms of only the quantities we know (i.e., $\widehat{a}$, ϵ, and δ, but *not* a.) [Hint: Break up the problem into two cases depending on $|a - \widehat{a}|$ and calculate the worst-case risk under either scenario.]

4. In this problem we will explore nearest neighbor classification in Python.

   The file `knn-example.py` provides a good start at this. You should be able to run this in the iPython environment simply by typing `run knn-example.py`. This uses the NumPy, Matplotlib, and scikit learn python packages. These should come included in the standard Anaconda distribution, but if you don't have them you will need to install them first.

   The file begins by loading the appropriate packages and fixing the random seed so that your results will be repeatable. It then generates a simple 2-dimensional dataset with n datapoints from two possible classes. Next it builds a k-nearest neighbor classifier. Finally, it plots the results. Before going further, spend some time with this and try to understand what the code is doing.

   In this problem I would like you to design a k-nearest neighbor classifier for several different values of n. In particular, I would like you to consider $n = 100, 500, 1000, 5000$. For each of these values of n, experiment with different choices of k and decide what the "best" choice of k is for each of these values of n (either based on the visual results, or using some quantitative method of your own devising). Provide a table showing your choices of k, and include a plot of the resulting classifier for each value of n.