CS 6350 Final Exam Review

1. What is the difference between the generalization error and the training error of classifiers? What is the fixed distribution assumption needed for PAC learning? How are online and batch learning different?

   - *Generalization Error*: Given a distribution $\mathcal{D}$ over examples, the *error* of a hypothesis h with respect to a target concept f is $err_{\mathcal{D}}(h) = Pr_{x\sim\mathcal{D}}[h(x) \neq f(x)]$.
   - *Training Error*: The error that is achieved when applying the model to the same data from which we trained on
   - *Empirical Error*: Contrast true error against emperical error. For a target concept f, the emperical error of a hypothesis h is defined for a training set S as the fraction of examples $x \in S$ for which functions $f \neq h$, denoted by $err_s(h)$.
   - The *Fixed Distribution Assumption* that is needed for PAC Learning is that we assume that the data in $\mathcal{D}$ is of a consistent distribution throughout the data and isn't built by multiple distributions
   - The difference between Online Learning and Batch Learning:

     Online Learning
     - No assumptions about the distribution of examples
     - Learning is a sequence of trials
       * Learner sees a single example, makes prediction. If there is a mistake, then update the hypothesis
     - *Goal:* To bound the total number of mistakes

     Batch Learning
     - Examples are drawn from a fixed (perhaps unknown) probability distribution $\mathcal{D}$ over the instance space
     - Learning uses a training set S, drawn from the distribution $\mathcal{D}$
     - *Goal:* Find hypothesis that has a low chance of making a mistake on a new example in $\mathcal{D}$

2. What are the formal definitions of PAC learnability and efficient PAC learnability?

   - Consider a concept class C defined over an instance space X (containing instances of length n), and a learner L using a hypothesis space H
     - The concept class C is *PAC Learnable* by L using H if:
       * for all $f \in C$
       * for all distribution $\mathcal{D}$ over X, and fixed $0 < \epsilon, \delta < 1$,
       * Given m examples sampled independently according to $\mathcal{D}$, the algorithm L produces, with probability at least $(1-\delta)$, a hypothesis $h \in H$ that has an error of at most ϵ
       * where m is *polynomial* in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, n, and $|H|$. Recall that

     $$err_{\mathcal{D}}(h) = Pr_{\mathcal{D}}[f(x) \neq h(x)]$$

     - The concept class C is *Efficiently PAC Learnable* if L can produce the hypothesis in time that is polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, n, and $|H|$
     - *Realistic Expectation of a Good Learner*: With a high probability, the learner will learn to close approximation to the target concept

3. What is the hypothesis space for various function classes – decision trees, monotone conjunctions and disjunctions, k-CNF's? Are they PAC learnable? Are they efficiently PAC learnable? (Use Occam's razor to show your answers)

   - Decision Trees:
     - For a decision tree, we assume that it contains binary features to not only simplify the math, but also provide a lower bound.

       Since we're using binary features, we can say that the size of our hypothesis space is

       $$|H| = 2^{2^n} \tag{1}$$

       We can use Occam's Razor by taking the logarithm of $|H|$ to see if our resulting PAC error would be polynomial bound, resulting in

       $$\ln|H| = 2^n \ln(2) \tag{2}$$

       which is not polynomial in n, rather it is exponential. Therefore, Decision Trees are *not* PAC learnable.

- Monotone Conjunctions and Disjunctions: These are a conjunction of literals and their negations, made up of n variables In order to test if it is PAC Learnable, we need to know the *Sample Complexity* using *Occam's Razor*. Occam's Razor favors smaller hypothesis spaces as we're able to learn the function with less samples
  - $|H| = 3^n$
  - $\log|H| = n\log(3) = \mathcal{O}(n)$ which is polynomial in n

$$m > \frac{1}{\epsilon}\left(\ln|H| - \ln(\delta)\right) \tag{3}$$

$$n > \frac{1}{\epsilon}\left(n\ln(3) - \ln(\delta)\right) \tag{4}$$

- k-CNF's: $(\ell_{1,1} \vee \ell_{1,2}, \vee \cdots \vee \ell_{1,k}) \wedge (\ell_{2,1} \vee \ell_{2,2}, \vee \cdots \vee \ell_{2,k}) \wedge \cdots$
  In order to solve this we need the *Sample Complexity*, that is if we had a consistent learner, how many examples will it need to guarentee PAC Learnability? To do so, we need to know the hypothesis space, *i.e.* how many k-CNF's are there?
  - Number of conjuncts $= \mathcal{O}\left((2n)^k\right)$
  - A k-CNF is a conjunction with these many variables
  - $|H|$ =Number of k-CNF's= $\mathcal{O}\left(2^{(2n)^k}\right)$
  - $\log|H| = \mathcal{O}\left(n^k\right)$ which is polynomial in n

4. Show that general Boolean functions are not PAC learnable.

- Number of Boolean Functions that exist with n variables: $|H| = 2^{2^n}$
- $\log|H| = 2^n\log(2) = \mathcal{O}(2^n)$ which is exponential in n, so general Boolean Functions are *not* PAC Learnable

5. What is agnostic learning? Is the training error for agnostic learning guarenteed to be zero?

- For a given data set, you're trying to learn a given concept f by using $h \in H$, but $f \notin H$
- With agnostic learning, you're not guarenteed that the training error will be zero
- *Goal*: Find a classifier $h \in H$ with a low training error

$$err_s(h) = \frac{|\{f(x) \notin h(x), x \in S\}|}{m} \tag{5}$$

  which defines the fraction of training examples that were misclassified. We *want* a guarentee that a hypothesis with a small training error will have a good accuracy on unseen examples

$$err_{\mathcal{D}}(h) = Pr_{x\sim\mathcal{D}}(f(x) \neq h(x)) \tag{6}$$

6. Derive the lower bounds for the number of examples needed for finite concept classes when the hypothesis class is consistent.

   - **Claim:** The probability $h \in H$ that:
     - Is consistent (yet bad) with m examples
     - Has error $err_{\mathcal{D}}(h) > \epsilon$

     is less than $|H|(1-\epsilon)^m$

     Proof: Let h be such a bad hypothesis with error $> \epsilon$. The probability that h is consistant with one exapmle is $Pr(f(x) = h(x)) < 1 - \epsilon$.

     The given training set consists of m examples drawn independently so, the probability that h is consistent with m examples $< (1-\epsilon)^m$. From this, the probability that a bad hypothesis $\in H$ is consistent with no examples is $|H|(1-\epsilon)^m$. We want to bound this and make it small, so we say

$$|H|(1-\epsilon)^m < \delta \tag{7}$$
$$\ln|H| + m\ln(1-\epsilon) < \ln(\delta) \tag{8}$$

     We know that the Taylor Series Approximation of $e^{-x} = 1 - x + \frac{x^2}{2!} - \cdots$, so we can use this to approximate $(1-\epsilon)$ with a first order Taylor Series Approximation which can be seen as

$$\ln|H| + m\ln\left(e^{-\epsilon}\right) < \ln(\delta) \tag{9}$$

     we can simplfiy the second term and move $\ln|H|$ to the right hand side of the equation

$$-\epsilon m \cdot \cancel{\ln(e)} < \ln(\delta) - \ln|H| \tag{10}$$

     In solving for the number of examples, m, we get

$$m > \frac{1}{\epsilon}(\ln|H| - \ln(\delta)) \tag{11}$$

     where the probability of a hypothesis $h \in H$ is consistent with a training set of size m is $(1-\delta)$ and will have an error $< \epsilon$ on future examples. This is called *Occam's Razor* because it expresses a preference towards smaller hypothesis spaces.

7. Derive the Hoeffding's bound to derive lower bound on sample complexity in the agnostic setting for a finite hypothesis class.

   - **Markov's Inequality**: Bounds the probability that a non-negative random variable exceedes a fixed value

$$P(x \geq a) \leq \frac{\langle x \rangle}{a} \tag{12}$$

   - **Chebyshev's Inequality**: Bounds the probability that a random variable differs from its expected value by more than a fixed number of standard deviations

$$P(|x - \mu| \geq k\sigma) \leq \frac{1}{k^2} \tag{13}$$

   - **Hoeffding's Inequality**: Gives the upper bounds on how much the sum of a set of random variables differs from its expected value

$$P(\mu > \bar{\mu} + \epsilon) \leq e^{-2m\epsilon^2} \tag{14}$$

     where μ is the expected mean, and $\bar{\mu}$ is the expirical mean (the mean over m trials). The empirical mean will not be too far from the expected mean if there are many samples. Hoeffding's Inequality also quantifies the convergence rate of this since it exponentially decays

Suppose we consider the true error (generalization error) $err_{\mathcal{D}}(h)$ to be a random variable. The training error over m examples is $err_s(h)$ which is the empirical estimate of this true error. To these, we can apply it to Hoeffding's Inequality

$$P(err_{\mathcal{D}}(h) > err_s(h) + \epsilon) \leq e^{-2m\epsilon^2} \tag{15}$$

$$err_{\mathcal{D}}(h) = Pr_{x\sim\mathcal{D}}(f(x) \neq h(x)) \tag{16}$$

$$err_s(h) = \frac{|\{f(x) \neq h(x), x \in S\}|}{m} \tag{17}$$

The probability that a single hypothesis h has a training error that is more than ϵ away from the true error is bounded in Equation (15). The learning algorithm looks for the best $h_i(x) \in |H|$ possible as its hypothesis. The probability that there is a hypothesis in $|H|$ whose training error is ϵ away from the true error is bounded by

$$P(\exists h; err_{\mathcal{D}}(h) > err_s(h) + \epsilon) \leq |H|\, e^{-2m\epsilon^2} \tag{18}$$

Where we want to bound our probability by some term δ

$$|H|\, e^{-2m\epsilon^2} \leq \delta \tag{19}$$

which we can take the logarithm of to simplify

$$\ln|H| - 2m\epsilon^2 \cdot \cancel{\ln(e)} \leq \ln(\delta) \tag{20}$$

in solving for m by subtracting $\ln|H|$ and dividing by $-2\epsilon^2$ gives

$$m \geq \frac{1}{2\epsilon^2}(\ln|H| - \ln(\delta)) \tag{21}$$

8. Deriving VC dimensions of various infinite hypothesis classes – bounded intervals, intervals, lines in 2 dimensions, axis parallel rectangles.

   - **Shattering**: A set of S examples is *shattered* by a set of functions H if for every position of the examples in S into positive and negative examples, there is a function in H that gives exactly these labels to these examples. *Intuition*: A rich set of functions shatters a large set of points
   - **VC Dimension**: The VC Dimension of a hypothesis space H over instance space X is the size of the largest *finite* subset of X that is shattered by H.

Concept Class	$\boldsymbol{VC(H)}$	**Why?**
Half Intervals	1	There is a dataset of size 1 that can be shattered. No dataset of size 2 can
Intervals	2	There is a dataset of size 2 that can be shattered. No dataset of size 3 can
Half-Spaces	3	There is a dataset of size 3 that can be shattered. No dataset of size 4 can
Linear Threshold Unit	$d+1$	
Neural Networks	# Parameters	
1-Nearest Neighbors	Infinite	

   - *Bounded Intervals*:
   - *Intervals*:
   - *Lines in 2 dimensions*:
   - *Axis Parallel Rectangles*:
   - **FINISH THIS ONE**

9. The theory of PAC learning upper bounds the true error of a classifier by two terms: true error $< A + B$. What are they?

   - An agnostic learner makes no commitment to whether $f \in H$ and returns the hypothesis with the least training error over at least m examples. It can guarentee with probability $(1 - \delta)$ that the training error is not off by more than ϵ from the true error if

$$m \geq \frac{1}{2\epsilon^2} \left(\ln |H| - \ln(\delta) \right) \tag{22}$$

   - *Generalization Bound*: How much the true error will deviate from the training error.

$$\underbrace{err_{\mathcal{D}}(h)}_{Expected\ Error} - \underbrace{err_s(h)}_{Training\ Error} \leq \sqrt{\frac{\ln |H| - \ln(\delta)}{2m}} \tag{23}$$

     which we can use to get an *upper bound* on the true error by manipulating Equation (23), resulting in

$$err_{\mathcal{D}}(h) \leq \sqrt{\frac{\ln |H| - \ln(\delta)}{2m}} + err_s(h) \tag{24}$$

10. What is a weak PAC algorithm?

    - **Boosting**: A general learning approach for constructing a *strong learner*, given a collection of (possibly infinite) weak learners ("rules of thumb")
    - **Ensemble Method**:
      - A class of learning algorithms that composes classifiers using other classifiers as building blocks
      - Boosing has stronger theoretical guarentees than other ensemble methods
    - **Strong PAC Algorithm**:
      - For any distribution over examples
      - For every $\epsilon > 0$, $\delta > 0$
      - Given a polynomial size of random examples
      - Finds a hypothesis with error $\leq \epsilon$ with probability $\geq (1 - \delta)$
    - **Weak PAC Algorithm**:
      - Same as a *Strong PAC Algorithm* except that $\epsilon > \frac{1}{2} - \gamma$

11. You are given the training examples in the table below. Suppose you use the following hypothesis space $H = \{sgn(x_1), sgn(x_2), -sgn(x_1), -sgn(x_2)\}$. That is learning requires picking a classifier that minimizes the error. Treating this your weak classifiers, step through three steps of the AdaBoost algorithm and write down the final hypothesis. Does it correctly classify the training set?

    The AdaBoost Algorithm can be seen in Algorithm 1. The norm Z_t is a normalization constant which ensures that the weights D_{t+1} add up to 1

(x_1, x_2)	y
$(1, 1)$	-1
$(1, -1)$	$+1$
$(-1, -1)$	-1
$(-1, -1)$	-1

    - The AdaBoost algorithm was implemented for 3 iterations below

Algorithm 1 AdaBoost($\{(\mathbf{x}_i, y_i)\}_m$)

Initialize $D_1(i) = \frac{1}{m}\ \forall\ i = \{1, 2, \ldots, m\}$
for $t = 1, 2, \cdots$ **to** T **do**
Find classifier h_t whose *weighted classificaiton error* $\epsilon_t(h)$ is better than chance:

$$\epsilon_t(h) = \frac{1}{2} - \frac{1}{2}\left(\sum_{i=1}^{m} D_t(i) y_i h(\mathbf{x}_i)\right)$$

Compute the norm of the *best* hypothesis Z_t as:

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Compute it's vote $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
Update values of weights for training examples:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \exp(-\alpha_t y_i h_t(\mathbf{x}_i))$$

end for
return $H_{final}(x) = \operatorname{sgn}\left(\sum_t \alpha_t h_t(\mathbf{x})\right)$

To initialize, make $D_1(i) = \frac{1}{4}$ since the set size is 4, then we can go on to calculating the *weighted classification error*

$$\epsilon_1(h_1) = \frac{1}{2} - \frac{1}{2}\left[\frac{1}{4}(-1)(1) + \frac{1}{4}(1)(1) + \frac{1}{4}(-1)(-1) + \frac{1}{4}(-1)(-1)\right] \tag{25}$$

$$= \frac{1}{2} - \frac{1}{2}[2] = \frac{1}{2} - \frac{1}{4} = \frac{1}{4} \tag{26}$$

$$\epsilon_1(h_2) = \frac{1}{2} - \frac{1}{2}\left[\frac{1}{4}(-1)(1) + \frac{1}{4}(1)(-1) + \frac{1}{4}(-1)(-1) + \frac{1}{4}(-1)(-1)\right] \tag{27}$$

$$= \frac{1}{2} - \frac{1}{2}\frac{1}{4}[2] = \frac{1}{4} \tag{28}$$

$$\epsilon_1(h_3) = \frac{1}{2} - \frac{1}{2}\left[\frac{1}{4}(-1)(-1) + \frac{1}{4}(1)(-1) + \frac{1}{4}(-1)(1) + \frac{1}{4}(-1)(1)\right] \tag{29}$$

$$= \frac{1}{2} - \frac{1}{2}\frac{1}{4}[-2] = \frac{3}{4} \tag{30}$$

$$\epsilon_1(h_4) = \frac{1}{2} - \frac{1}{2}\left[\frac{1}{4}(-1)(-1) + \frac{1}{4}(1)(1) + \frac{1}{4}(-1)(1) + \frac{1}{4}(-1)(1)\right] \tag{31}$$

$$= \frac{1}{2} - \frac{1}{2}\frac{1}{4}[1 + 1 - 1 - 1] = \frac{1}{2} \tag{32}$$

where the "winner" is $h_3(\mathbf{x})$ since it has the largest value of ϵ_t. We can use this to calculate Z_t, α_t

$$Z_1 = 2\sqrt{\epsilon_t(1-\epsilon_t)} = 2\sqrt{\frac{3}{4}\left(1 - \frac{3}{4}\right)} = 2\sqrt{\frac{3}{4}\cdot\frac{1}{4}} = \frac{\sqrt{3}}{2} \tag{33}$$

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) = \frac{1}{2}\ln\left(\frac{1-3/4}{3/4}\right) = \frac{1}{2}\ln\left(\frac{1/4}{3/4}\right) = -\frac{1}{2}\ln(3) \tag{34}$$

and we can update each value of $D_2(i)$ for the next iteration

$$D_2(1) = \frac{1/2}{\sqrt{3}} \exp\left(-\left(-\frac{1}{2}\right)\ln(3)(-1)(-1)\right) = \frac{1/2}{\sqrt{3}} \cdot 3^{1/2} = \frac{1}{2} \tag{35}$$

$$D_2(2) = \frac{1/2}{\sqrt{3}} \exp\left(-\left(-\frac{1}{2}\right)\ln(3)(1)(-1)\right) = \frac{1/2}{\sqrt{3}} \cdot 3^{-1/2} = \frac{1}{6} \tag{36}$$

$$D_2(3) = \frac{1/2}{\sqrt{3}} \exp\left(-\left(-\frac{1}{2}\right)\ln(3)(-1)(1)\right) = \frac{1/2}{\sqrt{3}} \cdot 3^{-1/2} = \frac{1}{6} \tag{37}$$

$$D_2(4) = \frac{1/2}{\sqrt{3}} \exp\left(-\left(-\frac{1}{2}\right)\ln(3)(-1)(1)\right) = \frac{1/2}{\sqrt{3}} \cdot 3^{-1/2} = \frac{1}{6} \tag{38}$$

$$\boxed{h_3(\mathbf{x}), \qquad \alpha_1 = -\frac{1}{2}\ln(3)} \tag{39}$$

Now we can move on to the next iteration, where we will first calculate the weighted error

$$\epsilon_2(h_1) = \frac{1}{2} - \frac{1}{2}\left[\frac{1}{2}(-1)(1) + \frac{1}{6}(1)(1) + \frac{1}{6}(-1)(-1) + \frac{1}{6}(-1)(-1)\right] \tag{40}$$

$$= \frac{1}{2} - \frac{1}{2}\left[\frac{3}{6} - \frac{1}{2}\right] = \frac{1}{2} \tag{41}$$

$$\epsilon_2(h_2) = \frac{1}{2} - \frac{1}{2}\left[\frac{1}{2}(-1)(1) + \frac{1}{6}(1)(-1) + \frac{1}{6}(-1)(-1) + \frac{1}{6}(-1)(-1)\right] \tag{42}$$

$$= \frac{1}{2} - \frac{1}{2}\frac{1}{4}\left[\frac{1}{6} - \frac{1}{2}\right] = \frac{4}{6} \tag{43}$$

$$\epsilon_2(h_3) = \frac{1}{2} - \frac{1}{2}\left[\frac{1}{2}(-1)(-1) + \frac{1}{6}(1)(-1) + \frac{1}{6}(-1)(1) + \frac{1}{6}(-1)(1)\right] \tag{44}$$

$$= \frac{1}{2} - \frac{1}{2}\frac{1}{4}\left[\frac{1}{2} - \frac{1}{2}\right] = \frac{1}{2} \tag{45}$$

$$\epsilon_2(h_4) = \frac{1}{2} - \frac{1}{2}\left[\frac{1}{2}(-1)(-1) + \frac{1}{6}(1)(1) + \frac{1}{6}(-1)(1) + \frac{1}{6}(-1)(1)\right] \tag{46}$$

$$= \frac{1}{2} - \frac{1}{2}\frac{1}{4}\left[\frac{1}{2} - \frac{1}{6}\right] = \frac{1}{3} \tag{47}$$

Where this time the maximum is from $h_2(\mathbf{x})$, so that is the hypothesis we're choosing

$$Z_2 = 2\sqrt{\epsilon_t(1-\epsilon_t)} = 2\sqrt{\frac{4}{6}\left(1 - \frac{4}{6}\right)} = 2\sqrt{\frac{4}{6}\cdot\frac{2}{6}} = \frac{2\sqrt{2}}{3} \tag{48}$$

$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) = \frac{1}{2}\ln\left(\frac{2/6}{4/6}\right) = -\frac{1}{2}\ln(2) \tag{49}$$

$$D_3(1) = \frac{3}{4\sqrt{2}} \exp\left(-\left(-\frac{1}{2}\right)\ln(2)(-1)(1)\right) = \frac{3}{4\sqrt{2}} \cdot 2^{-1/2} = \frac{3}{8} \tag{50}$$

$$D_3(2) = \frac{3}{12\sqrt{2}} \exp\left(-\left(-\frac{1}{2}\right)\ln(2)(1)(-1)\right) = \frac{3}{12\sqrt{2}} \cdot 2^{-1/2} = \frac{1}{8} \tag{51}$$

$$D_3(3) = \frac{3}{12\sqrt{2}} \exp\left(-\left(-\frac{1}{2}\right)\ln(2)(-1)(-1)\right) = \frac{3}{12\sqrt{2}} \cdot 2^{1/2} = \frac{1}{4} \tag{52}$$

$$D_3(4) = \frac{3}{12\sqrt{2}} \exp\left(-\left(-\frac{1}{2}\right)\ln(2)(-1)(-1)\right) = \frac{3}{12\sqrt{2}} \cdot 2^{1/2} = \frac{1}{4} \tag{53}$$

$$\boxed{h_2(\mathbf{x}), \qquad \alpha_2 = -\frac{1}{2}\ln(2)} \tag{54}$$

For the last iteration

$$\epsilon_3(h_1) = \frac{1}{2} - \frac{1}{2}\left[\frac{3}{8}(-1)(1) + \frac{1}{8}(1)(1) + \frac{1}{4}(-1)(-1) + \frac{1}{4}(-1)(-1)\right] \tag{55}$$

$$= \frac{1}{2} - \frac{1}{2}\left[\frac{1}{2} - \frac{1}{4}\right] = \frac{3}{8} \tag{56}$$

$$\epsilon_3(h_2) = \frac{1}{2} - \frac{1}{2}\left[\frac{3}{8}(-1)(1) + \frac{1}{8}(1)(-1) + \frac{1}{4}(-1)(-1) + \frac{1}{4}(-1)(-1)\right] \tag{57}$$

$$= \frac{1}{2} - \frac{1}{2}\frac{1}{4}\left[\frac{1}{2} - \frac{1}{2}\right] = \frac{1}{2} \tag{58}$$

$$\epsilon_3(h_3) = \frac{1}{2} - \frac{1}{2}\left[\frac{3}{8}(-1)(-1) + \frac{1}{8}(1)(-1) + \frac{1}{4}(-1)(1) + \frac{1}{4}(-1)(1)\right] \tag{59}$$

$$= \frac{1}{2} - \frac{1}{2}\frac{1}{4}\left[\frac{1}{4} - \frac{1}{2}\right] = \frac{5}{8} \tag{60}$$

$$\epsilon_3(h_4) = \frac{1}{2} - \frac{1}{2}\left[\frac{3}{8}(-1)(-1) + \frac{1}{8}(1)(1) + \frac{1}{4}(-1)(1) + \frac{1}{4}(-1)(1)\right] \tag{61}$$

$$= \frac{1}{2} - \frac{1}{2}\frac{1}{4}\left[\frac{4}{8} - \frac{1}{2}\right] = \frac{1}{2} \tag{62}$$

where we can see that $h_3(\mathbf{x})$ "won" again. Since we don't need to perform any more iterations, all we need to calculate is α_3 to get the final hypothesis

$$\alpha_3 = \frac{1}{2}\ln\left(\frac{3/8}{5/8}\right) = \frac{1}{2}\ln\left(\frac{3}{5}\right) \tag{63}$$

$$\boxed{h_3(\mathbf{x}), \qquad \alpha_3 = \frac{1}{2}\ln\left(\frac{3}{5}\right)} \tag{64}$$

To get the final Hypothesis, we need to sum the information in Equations (39, 54, 64)

$$H_{final}(\mathbf{x}) = \text{sgn}\left(-\frac{1}{2}\ln(3)h_3(\mathbf{x}) - \frac{1}{2}\ln(2)h_2(\mathbf{x}) - \frac{1}{2}\ln\left(\frac{3}{5}\right)\right) \tag{65}$$

$$= \text{sgn}\left(-\frac{1}{2}\left[\ln(5)h_3(\mathbf{x}) - \ln(2)h_2(\mathbf{x})\right]\right) \tag{66}$$

Now that we have our final hypothesis in Equation (66), we can test it against the given data to see it's performance. In doing so, the only mistake it makes is with the first point $\mathbf{x}_1$, which it classified as $+1$ instead of -1. Since it got all the other terms right, it had an empirical error of $\frac{3}{4}$.

12. What is the margin of a data set with respect to a hyperplane? Why does maximizing the margin improve this generalization?

- The *margin* of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it.
- Larger margins are better as it generalizes the data more and gives room for future examples that may be closer to the hyperplane. To maximize the margin, we use

$$\max_{\mathbf{w}} \min_{(\mathbf{x}_i, y_i)} \frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{||\mathbf{w}||} \tag{67}$$

where the term that is being minimized is representative of γ, the margin

13. What is the objective function for support vector machines? What is the objective function for regularized logistic regression?

- SVM Objective Objective Function:

$$\min_{\mathbf{w}} \underbrace{\frac{1}{2}\mathbf{w}^T\mathbf{w}}_{Regularization\ Term} + C\sum_i \max \underbrace{[0, 1 - y_i\mathbf{w}^T\mathbf{x}_i]}_{Loss\ Function} \tag{68}$$

- Logistic Regression Objective Function:

$$\min_{\mathbf{w}} \underbrace{\frac{1}{2}\mathbf{w}^T\mathbf{w}}_{Regularization\ Term} + C\sum_i \max \underbrace{\log\left[1+\exp(-y\mathbf{w}^T\mathbf{x}\right]}_{Loss\ Function} \tag{69}$$

- *Regularization Term*:
  - Maximize the margin
  - Imposes a preference over the hypothesis space and pushes for better generalization
  - Can be replaced with other regularization terms which impose other preferences
- *Empirical Loss*:
  - Hinge Loss Function
  - Penalizes weight vectors that make mistakes
  - Can be replaced with other loss functions which impose other preferences
- *Hyper Paramter*: C is a hyper parameter that controls the tradeoff between a large margin and a small hinge-loss

14. Derive the stochastic gradient descent algorithm for SVM and logistic regression.

- Stochastic Gradient Descent treats each individual data point in the data as the entire data set and takes the gradient of that one function. In order to do it for SVM and Logistic Regression, we need to utilize the correspoinding *loss functions* for each, from Equations (81, 83). What we want to do is solve the following optimization problem

$$\min_{\mathbf{w}} \left\{ \frac{\mathbf{w}^T\mathbf{w}}{2\sigma^2} + \sum_i L(h(\mathbf{x}_i), f(\mathbf{x}_i) \right\}$$

  where $L(h(\mathbf{x}_i), f(\mathbf{x}_i)$ is the loss function for the problem we are trying to solve. The stochastic gradient descent method updates the weight vector by going in the opposite direction of the gradient, so it will update as

$$\mathbf{w}_{(t+1)} = \mathbf{w}_{(t)} - r\vec{\nabla}J(\mathbf{w})$$

  where r is the "learning rate" of the function, *i.e.* how large of a stepsize it uses
- **SVM**: The function that we want to minimize is:

$$J(\mathbf{w}) = \min_{\mathbf{w}} \left\{ \frac{\mathbf{w}^T\mathbf{w}}{2\sigma^2} + \sum_i \max\left[0, 1 - y_i\mathbf{w}^T\mathbf{x}_i\right] \right\} \tag{70}$$

  which we can't take a direct derivative of because the loss function is not differentiable. The solution to this is to take a derivative of both cases of the max function and let the algorithm decide which to use at runtime. The first case is for when max chooses 0

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}\frac{\mathbf{w}^T\mathbf{w}}{2\sigma^2} = \frac{\partial}{\partial \mathbf{w}}\frac{||\mathbf{w}||^2}{2\sigma^2} \tag{71}$$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{||\mathbf{w}||}{\sigma^2} \tag{72}$$

  For max choosing $1 - y_i\mathbf{w}^T\mathbf{x}_i$ the result is, where we can also drop the summation since we want to take the gradient for a *single point*

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}}\frac{\mathbf{w}^T\mathbf{w}}{2\sigma^2} + 1 - y_i\mathbf{w}^T\mathbf{x} \tag{73}$$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{||\mathbf{w}||}{\sigma^2} - y\mathbf{x} \tag{74}$$

- **Logisitic Regression**: The function we want to minimize is:

$$J(\mathbf{w}) = \min_{\mathbf{w}} \left\{ \frac{\mathbf{w}^T\mathbf{w}}{2\sigma^2} + \sum_i \log\left(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)\right) \right\} \tag{75}$$

which unlike the SVM case *is* differentiable. We can remove the summation from this equation since we want to take the gradient of a single data point

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \left(\frac{\mathbf{w}^T\mathbf{w}}{2\sigma^2} + \log\left(1 + \exp(-y\mathbf{w}^T\mathbf{x})\right) \right) \tag{76}$$

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{||\mathbf{w}||}{\sigma^2} + \frac{-y\mathbf{x}}{1 + \exp(y\mathbf{w}^T\mathbf{x})} \tag{77}$$

15. Why does minimizing $||\mathbf{w}||$ improve generalization for a linearly separable dataset?

- Minimizing the norm prevents overfitting. This can be seen in Vapnik's Theorem on the generalization of margins. If we let H be the set of linear classifiers that separate the training set by a margin of at least γ, then we get

$$VC(H) \leq \min\left[\frac{R^2}{\gamma^2}, d\right] + 1 \tag{78}$$

with R representing the radius of the smallest sphere required to contain the data. This equation also implies that the larger the margin, the lower the VC dimension as well, where we've seen that the lower the VC dimension, the better the generalization. Thus, minimizing $||\mathbf{w}||$ maximizes the margin since

$$\gamma = \min_{(\mathbf{x}_i, y_i)} \frac{y_i(\mathbf{w}^T\mathbf{x} + b)}{||\mathbf{w}||} \tag{79}$$

This shows that $\gamma \propto \frac{1}{||\mathbf{w}||}$, which since γ is in the denominator in Vapnik's theorem, lowers the VC dimension, and thus improving generalization.

16. What is the logistic loss function?

- *Loss Function*: Should penalize mistakes and are minimizing the average loss over the training data
- The following is a list of common loss functions where *Perceptron Loss* is utilized in the Perceptron algorithm, *Hinge Loss* is utilized in SVM's, *Exponential Loss* is used in AdaBoost, and *Logistic Loss* is used in Logistic Regression

$$L_{Perceptron}(y, \mathbf{x}, \mathbf{w}) = \max\left[0, -y\mathbf{w}^T\mathbf{x}\right] \tag{80}$$

$$L_{Hinge}(y, \mathbf{x}, \mathbf{w}) = \max\left[0, 1 - y\mathbf{w}^T\mathbf{x}\right] \tag{81}$$

$$L_{Exponential}(y, \mathbf{x}, \mathbf{w}) = e^{-y\mathbf{w}^T\mathbf{x}} \tag{82}$$

$$L_{Logistic}(y, \mathbf{x}, \mathbf{w}) = \log\left(1 + e^{-y\mathbf{w}^T\mathbf{x}}\right) \tag{83}$$

where we can algo generalize our objective function as being

$$\min_{h \in H} \text{regularizer}(h) + C\frac{1}{m}\sum_i L\left(h(\mathbf{x}_i), f(\mathbf{x}_i)\right) \tag{84}$$

17. What is the difference between gradient descent and stochastic gradient descent?

- The differences between the two can be seen as follows:

Gradient Descent
- Update a set of parameters in an iterative manner to minimize a loss function
- Run through *all* of the samples in your training set to do a single update
- Error function is better minimized

Stoichastic Gradient Descent
- Update a set of parameters in an iterative manner to minimize a loss function
- Use *only one* training sample from your training set to do the update for a parameter at a particular iteration
- Usually converges much faster

18. Write down an expression for the optimum weight vector for an SVM in terms of training examples.

    - Let $\mathbf{w}$ be the minimizer of the SVM problem from some dataset with m examples $\{(\mathbf{x}_i, y_i)\}_m$. Then for $i = \{1, 2, \ldots, m\}$, there must exist some parameter α_i such that the *optimum* $\mathbf{w}$ can be built. This can be expressed mathematically as

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

19. Derive the kernel perceptron algorithm.

    - This can be derived in the following way:

      The perceptron algorithm only updates everytime it makes a mistake. To update the weight vector, it updates as $\mathbf{w}_{i+1} = \mathbf{w}_i + y_i \mathbf{x}_i$. If we have a function $\phi(\mathbf{x})$ that maps $\mathbf{x}$ to a new feature space, we have $\mathbf{w}_{i+1} = \mathbf{w}_i + y_i \phi(\mathbf{x}_i)$. If the perceptron algorithm makes M errors on the training data, the final weight vector can be represented as

$$\mathbf{w} = \sum_{i:(\mathbf{x}_i, y_i) \in M} y_i \phi(\mathbf{x}_i) \tag{85}$$

      On top of this, we also know that the classification of the perceptron algorithm comes from

$$y = \operatorname{sgn}\left(\mathbf{w}^T \mathbf{x}\right) = \operatorname{sgn}\left(\mathbf{w}^T \phi(\mathbf{x})\right) \tag{86}$$

      which can be generalized by plugging in the definition of $\mathbf{w}$ given above

$$y = \operatorname{sgn}\left(\sum_{i:(\mathbf{x}_i, y_i) \in M} y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})\right) \tag{87}$$

      we also know that the kernel function is defined as

$$\kappa(\mathbf{x}, \mathbf{z}) \equiv \phi(\mathbf{x})^T \phi(\mathbf{z}) \tag{88}$$

      which we can use in our new definition of the classification, giving us

$$y = \operatorname{sgn}\left(\sum_{i:(\mathbf{x}_i, y_i) \in M} y_i \kappa(\mathbf{x}_i, \mathbf{x})\right) \tag{89}$$

Algorithm 2 KernelPerceptron($\{(\mathbf{x}_i, y_i)\}_n$)

```
w ← 0
for i = 1 to M do
  if y ≠ sgn(Σ_{x_i, y_i ∈ M} y_i κ(x, x_i)) then
    w ← w ∪ (x_i, y)
  end if
end for
return sgn(Σ_{x_i, y_i ∈ M} y_i κ(x, x_i))
```

20. For two vectors $\mathbf{x}$ and $\mathbf{z}$, are the following functions valid kernels?

- A kernel is valid if and only if, it is symmetric $\kappa(\mathbf{x}, \mathbf{z}) = \kappa(\mathbf{z}, \mathbf{x})$, and if it is positive-semi definite $\left(\sum_{j=1}^{n}\sum_{i=1}^{n} c_i c_j \kappa(\mathbf{x}_i, \mathbf{x}_j)\right) \geq 0$. From these definitions, it also follows that $\kappa(\mathbf{x}, \mathbf{z})$ is valid if and only if there exists a function $\kappa(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T \cdot \phi(\mathbf{z})$
- $\mathbf{x}^T\mathbf{z}$
  - This is trivially shown as true. If $\kappa(\mathbf{x}, \mathbf{z}) : \phi(\mathbf{x}) \to \mathbf{x}$, then we have a function mapping it onto itself. The resutling dot product that would form the question would be $\phi(\mathbf{x}^T) \cdot \phi(\mathbf{z})$, so it is trivially true that it is a valid mapping, thus making $\mathbf{x}^T\mathbf{z}$ a valid kernel.
- $\left(1 + \mathbf{x}^T\mathbf{z}\right)^2$

  If we expand this equation we get

$$\kappa(\mathbf{x}, \mathbf{z}) = 1^2 + 2\sum_{i=1}^{n} x_i z_i + \left(\sum_{i=1}^{n} x_i z_i\right)^2 \tag{90}$$

  with, without loss of generality, can be rewritten as

$$\kappa(\mathbf{x}, \mathbf{z}) = 1 + 2\mathbf{x}^T\mathbf{z} + \left|\left|\mathbf{x}^T\mathbf{z}\right|\right|^2 \tag{91}$$

  which can be brought into a *mapping* to $\phi(\mathbf{x})$ such as:

$$\phi(\mathbf{x}) = \left(1, \sqrt{2} \cdot \mathbf{x}, \mathbf{x}^2\right) \Rightarrow \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \tag{92}$$

  which is a valid kernel $\kappa(\mathbf{x}, \mathbf{z})$
- $\exp\left(||\mathbf{x}|| \cdot ||\mathbf{z}|| \cdot (12 + \mathbf{x}^T\mathbf{z})^{17}\right) + \left(\mathbf{x}^T\mathbf{z}\right)^3$

  This can be shown by example that this is not a valid kernel in the following way. We can build a matrix $\mathbf{M}$ from our function, and we have to show that it is not positive semi-definite, by $\mathbf{b}^T\mathbf{M}\mathbf{b}$, where without loss of generality we can define $\mathbf{b}$ as being the identity vector such that $\mathbf{b} \in \mathbb{1}^{n\times 1}$. Following this, we can define our $\mathbf{x} = \mathbb{1}^{n\times 1}$ and $\mathbf{z} = (-2)\mathbb{1}^{n\times 1}$. Using this information, to show it's not positive semi-definite, we essentially need to calculate the above "kernel" and show that it is less than zero. In doing so

$$\exp\left(\cancel{||\mathbf{x}||} \cdot \sqrt{(-2)^2 n} \cdot \left(\sum_{i=1}^{n} x_i z_i\right)^{17}\right) + \left(\sum_{i=1}^{n} x_i z_i\right)^3 \tag{93}$$

$$\exp\left(2\sqrt{n} \cdot (-2n)^{17}\right) + (-2n)^3 \tag{94}$$

$$\exp\left(-2^{18} n^{17/2}\right) - 8n^3 < 0 \tag{95}$$

  The last statement holds true since the exponential function decays quickly and for any sufficient n it will overcome the exponential and make the entire equation negative.

21. When the training error is low and the test error is high, is it a symptom of high bias or high variance?

- **Bias**: The true error (loss) of the *best* predictor in teh hypothesis set.
  - What will the bias be if the hypothesis set can not represent the target function? (high or low?)
    - Bias will be non-zero, possiby high
  - *Underfiting*: When bias is high
- **Variance**: Describes how much the *best* classifier depends on the training set
  - *Overfitting*: High variance
  - Increases when classifiers become more complex
  - Decreases with larger training sets
- **Error**: $bias + variance$ $(+ noise)$

- *High bias*: Both training and test set error can be high
  - Arises when the classifier can not represent the data
- *High variance*: Training error can be low, but the test error will beh high
  - Arises when the learner overfits the training set

22. What does the regularization term for regularized loss minimization do with respect to the bias variance tradeoff?

    - The regularization term helps reduce the variance and makes the model more generalized. For example, if we have a feature set for a single example $\mathbf{x}_i$ as being

      $$\mathbf{x}_i = (x_1, x_2, \ldots, x_{100})^T \tag{96}$$

      then it's clear that if $\mathbf{w}$ is unbounded, it can fit any data point successfully. However, if the weight vector updated every time it saw a data point to "match" the new data point's weight vector, no data set would ever be able to be learned as the weightvector would vary widely.
    - **Managing of bias and variance**:
      - *Ensemble Methods* reduce variance
        - Multiple classifiers are combined (*i.e.* boosting)
      - *k-Nearest Neighbors*
        - Increasing k generally increases bias, reduces variance
      - *Decision Trees of a given depth*
        - Increasing depth decreases bias, increases variance
      - *SVM's*
        - Higher degree polynomial kernels decreases bias, increases variance
        - Stronger regularization increases bias, decreases variance

23. What is the difference between MAP learning and MAP prediction? Give an example of a binary classifier that is learned using the MAP principle and uses MAP prediction.

    - **Maximum a posteriori (MAP) Learning**:

      $$h_{MAP} = \arg\max_{h \in H} P(\mathcal{D}|h)P(h) \tag{97}$$

      - Count how often features occur with each label. Normalize to get likelihoods
      - Priors from fraction of examples with each label
      - Generalizes to multiclass
    - **Maximum a posteriori (MAP) Prediction**:

      $$h = \arg\max_{y} P(X = \mathbf{x}|Y = y)P(Y = y) \tag{98}$$

      - Use learned probabilities to find the highest scoring label
    - In essence, MAP learning is Maximum A Posteriori learning, which tries to find the hgihest posterior probability of a given hypothesis for some data. MAP prediction attempts to solve the problem of maximizing the posterior probability of the predictions given the data.
    - The following is a biniary classification example that uses MAP Prediction. It utilizes information in Table 1 and Table 2. The question is, based on that data, if todays $Temperature = Hot(H)$ and $Wind = Weak(W)$, should I play tennis today?

Table 1: Prior Values

Play Tennis	$\boldsymbol{P}$**(play tennis)**
Yes	0.30
No	0.70

Table 2: Likelihood Values

Temperature	Wind	$P(T, W \mid Tennis = Yes)$
Hot	Strong	0.15
Hot	Weak	0.40
Cold	Strong	0.10
Cold	Weak	0.35
Temperature	**Wind**	$P(T, W \mid Tennis = No)$
Hot	Strong	0.40
Hot	Weak	0.10
Cold	Strong	0.30
Cold	Weak	0.20

$$h(H, W) = \arg\max_y P(H, W|play?)P(play?) \tag{99}$$

$$P(H, W|Yes)P(Yes) = 0.40 \times 0.30 = 0.12 \tag{100}$$

$$P(H, W|No)P(No) = 0.10 \times 0.70 = 0.07 \tag{101}$$

MAP Prediction: Yes, you should play tennis today

24. Show that the least squares estimation is equivalent to maximum likelihood estimation for a specific probabilistic model.

- **Maximum Likelihood Hypothesis**:

$$h_{ML} = \arg\max_{h \in H} P(\mathcal{D}|h) \tag{102}$$

$$\tag{103}$$

Suppose H consists of real valued functions and its inputs are vectors $\mathbf{x} \in \mathbb{R}^{d \times 1}$ and output $y \in \mathbb{R}$. The training data was generated as follows:

  - Input $\mathbf{x}_i$ is drawn uniformly at random
  - True function $f(\mathbf{x})$ is applied to get $f(\mathbf{x}_i)$
  - The value is perturbed by some noise e_i: $y_i = f(\mathbf{x}_i) + e_i$
  - There are m training examples $(\mathbf{x}_i, y_i)$ that are generated in this process

- Recall that a normal distribution is parameterized by its mean μ and variance σ. If h was a true function, then the mean of y_i would be $h(\mathbf{x}_i)$, so we have the probability density function of

$$P(y_i|h, \mathbf{x}_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i - h(\mathbf{x}_i)^2}{2\sigma^2}\right) \tag{104}$$

which represents the probability of observing one data point $(\mathbf{x}_i, y_i)$ if it were generated by the function $h(\mathbf{x}_i)$. Each example in our data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ is generated independently in this process, so we can use the Bayes Assumption which states

$$P(\mathcal{D}|h) = \prod_{i=1}^{m} P(y_i, \mathbf{x}_i|h) \tag{105}$$

our goal is to find the maximum likelihood hypothesis

$$h_{ML} = \arg\max_{h \in H} P(\mathcal{D}|h) = \arg\max_{h \in H} \prod_{i=1}^{m} P(y_i|h, \mathbf{x}_i) \tag{106}$$

$$= \arg\max_{h \in H} \prod_{i=1}^{m} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i - h(\mathbf{x}_i))^2}{2\sigma^2}\right) \tag{107}$$

where we can take the logarithm to simplify since the logarithm is an increasing function and it won't effect our maximization

$$h_{ML} = \arg\max_{h\in H} \log\left(\prod_{i=1}^{m} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i - h(\mathbf{x}_i))^2}{2\sigma^2}\right)\right) \tag{108}$$

$$= \arg\max_{h\in H} \sum_{i=1}^{m} \log\left(\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i - h(\mathbf{x}_i))^2}{2\sigma^2}\right)\right) \tag{109}$$

$$= \arg\max_{h\in H} \sum_{i=1}^{m} \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \log\left[\exp\left(-\frac{(y_i - h(\mathbf{x}_i))^2}{2\sigma^2}\right)\right] \tag{110}$$

$$= \arg\max_{h\in H} \sum_{i=1}^{m} \left(-\log\left(\sigma\sqrt{2\pi}\right)\right) - \frac{(y_i - h(\mathbf{x}_i))^2}{2\sigma^2} \tag{111}$$

where we can remove the first term and the $2\sigma^2$ term since they are constants and are independent of the maximization, resulting in

$$h_{ML} = \arg\max_{h\in H} \sum_{i=1}^{m} \left(-(y_i - h(\mathbf{x}_i))^2\right) \tag{112}$$

which gives the hypothesis of

$$h_{ML} = \arg\min_{h\in H} \sum_{i=1}^{m} (y_i - h(\mathbf{x}_i))^2 \tag{113}$$

where for linear functions we consider $h(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i$

$$h_{ML} = \arg\min_{h\in H} \sum_{i=1}^{m} \left(y_i - \mathbf{w}^T\mathbf{x}_i\right) \tag{114}$$

which is the probabilistic version of least squares regression

25. What is the difference between Bayes optimal classification and prediciton according to the MAP classifier? Suppose you have a learning task. Your hypothesis class H contains a countably infinite number of functions $(h_1, h_2, h_3, \ldots)$. It so happens that for a particular dataset $\mathcal{D}$, the posterior probabilities of these functions are:

$$P(h_1|\mathcal{D}) = \frac{1}{2} - \epsilon$$
$$P(h_2|\mathcal{D}) = \frac{1}{4} + \epsilon$$
$$P(h_i|\mathcal{D}) = \frac{1}{2^i} \quad \forall\ i > 2$$

What is the MAP hypothesis for this dataset?

- The equation for h_{MAP} prediction is as follows

$$h_{MAP} = \arg\max_{h\in H} P(\mathcal{D}|h)P(h); \qquad P(\mathcal{D}|h) = \frac{P(h|\mathcal{D})P(\mathcal{D})}{P(h)} \tag{115}$$

$$= \arg\max_{h\in H} \frac{P(h|\mathcal{D})P(\mathcal{D})}{P(h)} P(h) = \arg\max_{h\in H} P(h|\mathcal{D})P(\mathcal{D}) \tag{116}$$

which for a set of data with a uniform random sampling becomes

$$h_{MAP} = \arg\max_{h\in H} P(h|\mathcal{D}) \tag{117}$$

Using this, we can find the maximum hypothesis that will be given, which turns out to be a joint set that is dependent upon the value of ϵ. These two results can easily be seen since $\max_{i>2} h_i = \frac{1}{8}$, which would not be anywhere near h_1 and h_2. The value of epsilon was found by equating $h_1 = h_2$ and solving for it. An assumption that is made is that the algorithm will choose h_i with the smallest i in order to break a "draw" (more than one hypothesis have the same value).

$$h_{MAP} = \begin{cases} h_1 & = & \frac{1}{2} - \epsilon & \left(\epsilon \leq \frac{1}{8}\right) \\ h_2 & = & \frac{1}{4} + \epsilon & \left(\epsilon > \frac{1}{8}\right) \end{cases} \tag{118}$$

Now, for a new example $\mathbf{x}$, the first function h_1 predicts the label $+1$ and all the others predict -1. That is, $P(y = -1|h_i) = 1$ for all h_i except for the first one and $P(y = +1|h_1) = 1$.

Do the predictions of the MAP hypothesis and the Bayes optimal classifier agree?

- **MAP Hypothesis**:
  - This is relatively straight forward. In the question, it states that h_1 predicts $+1$ and -1 for the rest, so depending on the value of ϵ, if $\epsilon \leq \frac{1}{8}$, then it will predict $+1$, and if $\epsilon > \frac{1}{8}$, it will predict -1.
- **Bayes Optimal Classifier**:
  - Bayes Optimal Classifier is defined as the following

$$h(y) = \arg\max_y \sum_{h_i \in H} P(y|h_i) P(h_i|\mathcal{D}) \tag{119}$$

The problem also states the probabilities for $P(y|h_i)$ which can be used to get the solution. To do so, we need to test *two* cases, for $y = 1$ and $y = -1$. First, for $y = 1$

$$h(y) = \sum_{h_i \in H} P(y = 1|h_i) P(h_i|\mathcal{D}) \tag{120}$$

$$= \underbrace{P(y = 1|h_1)}_{1} P(h_1|\mathcal{D}) + \sum_{i>1} \underbrace{P(y = 1|h_i)}_{0} P(h_i|\mathcal{D}) \tag{121}$$

$$= \frac{1}{2} - \epsilon \tag{122}$$

For $y = -1$

$$h(y) = \sum_{h_i \in H} P(y = -1|h_i) P(h_i|\mathcal{D}) \tag{123}$$

$$= \underbrace{P(y = -1|h_1)}_{0} P(h_1|\mathcal{D}) + \underbrace{P(y = -1|h_2)}_{1} P(h_2|\mathcal{D}) + \sum_{i=3}^{\infty} \underbrace{P(y = -1|h_i)}_{1} P(h_i|\mathcal{D}) \tag{124}$$

$$= P(h_2|\mathcal{D}) + \sum_{i=3}^{\infty} P(h_i|\mathcal{D}) \tag{125}$$

$$= \frac{1}{4} + \epsilon + \underbrace{\sum_{i=3}^{\infty} \frac{1}{2^i}}_{\text{converges to 2}} \tag{126}$$

$$= \frac{1}{4} + \epsilon + 2 \tag{127}$$

where we again need to check for what values of ϵ one is chosen over the other. In doing so, the classification of $\mathbf{x}$ is given as

$$h(y) = \begin{cases} y = & 1 & \left(\frac{1}{2} - \epsilon\right) & \left(\epsilon \leq -\frac{7}{4}\right) \\ y = & -1 & \left(\frac{1}{4} + \epsilon + 2\right) & \left(\epsilon > -\frac{7}{4}\right) \end{cases} \tag{128}$$

  - The Bayes Optimal Classifier would predict $+1$ for $\epsilon \leq -\frac{7}{4}$ and -1 for $\epsilon > -\frac{7}{4}$.

- We can't say for sure whether they agree or not since we don't know the value of ϵ.

26. What is the Naïve Bayes assumption?

    - What if all of the features are conditionally indpendent, then what does the hypothesis function look like? That is

    $$P(x_1, x_2, \ldots, x_d|y) = P(x_1|y)P(x_2|y)\cdots P(x_d|y) \tag{129}$$

    Requires only d number for each label. kd features overall! So if the features are conditionally independent given the label y, to predict we need to sets of probabilities: (1) Prior $P(y)$ and (2) For each $\mathbf{x}_i$, we have the likelihood $P(\mathbf{x}_i|y)$. If we have these, we can make the decision rule as:

    $$h_{NB}(\mathbf{x}) = \arg\max_y P(y)P(x_1, x_2, \ldots, x_d|y) \tag{130}$$

    $$= \arg\max_y P(y)\prod_{i=1}^{d} P(x_i|y) \tag{131}$$

27. Derive the maximum likelihood Naïve Bayes classifier when the conditional probabilities of each feature given the label is a Bernoulli trial and the prior is also a Bernoulli trial.

    - **Maximum Likelihood Estimation**: Given a dataset $\{(\mathbf{x}_i, y_i)\}_m$ which is composed of data that is independent and uniformly distributed.

    $$h_{ML} = \arg\max_{h\in H} P(\mathcal{D}|h) \tag{132}$$

    where h is all probabilities used to construct the Naïve Bayes decision. Since the data is independent and uniformly distributed, we can rewrite by using the Bayes assumption

    $$h_{ML} = \arg\max_{h\in H} \prod_{i=1}^{m} P\left((\mathbf{x}_i, y_i)|h\right) \tag{133}$$

    $$= \arg\max_{h\in H} \prod_{i=1}^{m} P(\mathbf{x}_i|y_i, h)P(y_i|h) \tag{134}$$

    $$= \arg\max_{h\in H} \prod_{i=1}^{m} P(y_i|h)\prod_j P(\mathbf{x}_{i,j}|y_i, h) \tag{135}$$

    where j represents the j^{th} feature of $\mathbf{x}_i$. We can take the logarithm of this to simplify

    $$h_{ML} = \arg\max_{h\in H} \log\left[\prod_{i=1}^{m} P(y_i|h)\prod_j P(\mathbf{x}_{i,j}|y_i, h)\right] \tag{136}$$

    $$= \arg\max_{h\in H} \log\left[\prod_{i=1}^{m} P(y_i|h)\right] + \log\left[\prod_{i=1}^{m}\prod_j P(\mathbf{x}_{i,j}|y_i, h)\right] \tag{137}$$

    $$= \arg\max_{h\in H} \sum_{i=1}^{m} \log\left[P(y_i|h)\right] + \sum_{i=1}^{m}\sum_j \log\left[P(\mathbf{x}_{i,j}|y_i, h)\right] \tag{138}$$

    to simplify this problem, we can assume that there are only two labels $[1, 0]$ and the features are binary. We can therefore represent each example as a Bernoulli trial, where below builds our *prior* and *likelihood* parameters, where the likelihood is for each feature of a given label $\mathbf{x}_i$

    Prior:

    $$P(y = 1) = p \tag{139}$$

    $$P(y = 0) = 1 - p \tag{140}$$

where, since h consists of all hypotheses, we can say

$$P(y_i|h) = p^{[y_i=1]}(1-p)^{[y_i=0]} \tag{141}$$

where $[z]$ is the indicator function which has a value of 1 if true, 0 else

Likelihood:

$$P(x_j = 1|y = 1) = a_j; \quad P(x_j = 0|y = 1) = (1 - a_j) \tag{142}$$

$$P(x_j = 1|y = 0) = b_j; \quad P(x_j = 0|y = 0) = (1 - b_j) \tag{143}$$

we can define

$$P(x_{i,j}|y_i, h) = P(y_i|h)P(x_{i,j}|y_i) \tag{144}$$

$$= a_j^{[y_i=1,x_{i,j}=1]}(1 - a_j)^{[y_i=1,x_{i,j}=0]} \times b_j^{[y_i=0,x_{i,j}=1]}(1 - b_j)^{[y_i=0,x_{i,j}=0]} \tag{145}$$

we can substitute these values into Equation (138), giving us

$$h_{ML} = \arg\max_{h\in H} \sum_{i=1}^{m} \log\left[P(y_i|h)\right] + \sum_{i=1}^{m}\sum_{j} \log\left[P(x_{i,j}|y_i, h)\right] \tag{146}$$

$$h_{ML} = \arg\max_{h\in H} \sum_{i=1}^{m} \log\left[p^{[y_i=1]}(1-p)^{[y_i=0]}\right] \tag{147}$$

$$+ \sum_{i=1}^{m}\sum_{j} \log\left[a_j^{[y_i=1,x_{i,j}=1]}(1 - a_j)^{[y_i=1,x_{i,j}=0]} \times b_j^{[y_i=0,x_{i,j}=1]}(1 - b_j)^{[y_i=0,x_{i,j}=0]}\right] \tag{148}$$

we can maximize this by taking the derivative with respect to $p,\ a_j,\ b_j$ and setting equal to zero so we can maximize it. Solving that gives us the updates for $p,\ a_j,\ b_j$, which are

$$p = \frac{\text{Count}\,(y = 1)}{\text{Count}\,(y = 1) + \text{Count}\,(y = 0)} \tag{149}$$

$$a_j = \frac{\text{Count}\,(y = 1, x_{i,j} = 1)}{\text{Count}\,(y = 1)} \tag{150}$$

$$b_j = \frac{\text{Count}\,(y = 0, x_{i,j} = 1)}{\text{Count}\,(y = 0)} \tag{151}$$

28. Derive the maximum likelihood estimate for Naïve Bayes when the features are assumed to be drawn from a normal distribution.

- This model assumes the probability of the i^{th} feature taking value x_i, and when label 1 is defined by a normal distribution with the mean of $\mu_{1,i}$

$y = 1$

$$P(x_i|y = 1) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_{1,i})}{2\sigma^2}\right) \tag{152}$$

$y = 0$

$$P(x_i|y = 0) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i - \mu_{0,i})}{2\sigma^2}\right) \tag{153}$$

where we get

$$P(x_i|y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-[y = 1]\frac{(x_i - \mu_{1,i})}{2\sigma^2}\right) \exp\left(-[y = 0]\frac{(x_i - \mu_{0,i})}{2\sigma^2}\right) \tag{154}$$

with $[y = 1]$ and $[y = 0]$ are our indicator functions. Simplifying this expression

$$P(x_i|y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-[y = 1]\frac{(x_i - \mu_{1,i})}{2\sigma^2} - [y = 0]\frac{(x_i - \mu_{0,i})}{2\sigma^2}\right) \tag{155}$$

If $P(y=1)=p$, we can now use the Naïve Bayes assumption to write the probability of a labeled example $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathbb{R}^{n\times 1}$, and the dataset $S = \{(\mathbf{x}_i, y_i)\}$

$$P(\mathbf{x}_i, y) = p^{[y_i=1]}(1-p)^{[y_i=0]} \prod i = 1^m \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-[y=1]\frac{(x_i-\mu_{1,i})}{2\sigma^2} - [y=0]\frac{(x_i-\mu_{0,i})}{2\sigma^2}\right) \tag{156}$$

where if we take the logarithm to simplify, we get

$$\log(P(\mathbf{x}_i, y)) = [y=1]\log(p) + [y=0]\log(1-p) - \log(\sigma\sqrt{2\pi}) \tag{157}$$

$$- \sum_{i=1}^{m} [y=1]\frac{(x_i-\mu_{1,i})}{2\sigma^2} + [y=0]\frac{(x_i-\mu_{0,i})}{2\sigma^2} \tag{158}$$

where we can take the derivative with respect to $p,\ \mu_{1,i},\ \mu_{0,i}$ to get the updates, which are

$$p = \frac{\text{number of examples with label 1}}{\text{total number of examples}} \tag{159}$$

$$\mu_{1,i} = \frac{\text{mean of all the } i^{th} \text{ features for label 1}}{\text{total number of examples with label 1}} \tag{160}$$

$$\mu_{0,i} = \frac{\text{mean of all the } i^{th} \text{ features for label 0}}{\text{total number of examples with label 0}} \tag{161}$$

29. What is the difference between a discriminative and generative classifier?

Discriminative Classifiers
- *Goal*: To learn how to make predictions
- Look at many postivie and negative examples
- Discover regularities in the data
- Use these to construct a prediction policy
- Assumtions come in the form of the hypothesis class
- *General*: Approximating $h : X \to Y$ is estimating $P(Y|X)$

Generative Classifiers
- Explicitely model how instances in each category are generated, that is learn $P(X|Y)$ and $P(Y)$
- We did this for Naïve Bayes (Naïve Bayes generation model)
- Predict $P(Y|X)$ using Bayes Rule

General: Generative classifiers learn a model of the joint probability $P(\mathbf{x}, y)$ of the inputs $\mathbf{x}$ and y, and make their predictions by using Bayes rules to calculate $P(y|\mathbf{x})$. Discriminative classifiers model the posterior $P(y|x)$ directly.

Bayes Rule:

$$P(Y=y|X=\mathbf{x}) = \frac{P(X=\mathbf{x})P(Y=y)}{P(X=\mathbf{x})}$$

30. For logistic regression, show that a particular choice of the prior for MAP learning is equivalent to using a squared norm regularizer on the weight vector.

- Hypothesis of logistic regression: All functions are of the form

$$h_{\mathbf{w}}(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})} \tag{162}$$

where $\sigma(z)$ is the sigmoid function defined as

$$\sigma(z) = \frac{1}{1-\exp(-z)} \tag{163}$$

Using this, we can build our probabilities as

$$P(y=1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1-\exp(-\mathbf{w}^T\mathbf{x})} \tag{164}$$

$$P(y=0|\mathbf{x}) = 1-\sigma(\mathbf{w}^T\mathbf{x}) = 1 - \frac{1}{1-\exp(-\mathbf{w}^T\mathbf{x})} \tag{165}$$

$$= \frac{1}{1+\exp(\mathbf{w}^T\mathbf{x})} \tag{166}$$

we can generalize this to a single equation for *any* value of y, which would be

$$P(y_i|\mathbf{x}) = \frac{1}{1+\exp(-y_i\mathbf{w}^T\mathbf{x})} \tag{167}$$

The idea behind this is if $P(y_i|\mathbf{x}) > .5$, label as $+1$, else label as -1. To predict: $\text{sgn}\left(\mathbf{w}^T\mathbf{x}\right)$. For this problem, we have some training data $S = \{(\mathbf{x}_i, y_i)\}_m$ and we want to find $\mathbf{w}$ such that $P(S|\mathbf{w})$ is maximized. We know our examples are independent and uniformly distributed, so we can use Bayes Assumption

$$h = \arg\max_{\mathbf{w}} P(S|\mathbf{w}) = \arg\max_{\mathbf{w}} \prod_{i=1}^{m} P(y_i|\mathbf{x}_i, \mathbf{w}) \tag{168}$$

Take the logarthim to help simplify

$$= \arg\max_{\mathbf{w}} \log\left(\prod_{i=1}^{m} P(y_i|\mathbf{x}_i, \mathbf{w})\right) \tag{169}$$

$$= \arg\max_{\mathbf{w}} \sum_{i=1}^{m} \log\left(P(y_i|\mathbf{x}_i, \mathbf{w})\right) \tag{170}$$

Plugging in $P(y_i|\mathbf{x}, \mathbf{w})$ from Equation (167)

$$h = \max_{\mathbf{w}} \sum_{i=1}^{m} \log\left[\frac{1}{1+\exp(-y_i\mathbf{w}^T\mathbf{x})}\right] \tag{171}$$

$$= \min_{\mathbf{w}} \sum_{i=1}^{m} \log\left[1+\exp(-y_i\mathbf{w}^T\mathbf{x})\right] \tag{172}$$

Adding prior to the weights: Suppose each weight vector is drawn independently from a normal distribution with a mean of zero and a standard deviation of σ^2, we can build the prior for the weight vector as

$$P(\mathbf{w}) = \prod_i P(w_i) = \prod i \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{w_i}{\sigma^2}\right) \tag{173}$$

where we learn by solving

$$\max_{\mathbf{w}} P(S|\mathbf{w})P(\mathbf{w}) \tag{174}$$

$$\max_{\mathbf{w}} \prod_{i=1}^{m} P(y_i|\mathbf{x}_i, \mathbf{w}) \prod_j \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{w_i}{\sigma^2}\right) \tag{175}$$

Taking the logarithm and removing constants since they're independent of the max

$$\max_{\mathbf{w}} \sum_{i=1}^{m} \log\left[P(y_i|\mathbf{x}_i, \mathbf{w})\right] - \sum_{j=1}^{d}\left(-\left(\frac{w_i}{\sigma}\right)^2\right) \tag{176}$$

$$\max_{\mathbf{w}} \sum_{i=1}^{m} -\log\left[1+\exp(-y_i\mathbf{w}^T\mathbf{x})\right] - \frac{1}{\sigma^2}\mathbf{w}^T\mathbf{w} \tag{177}$$

$$\min_{\mathbf{w}} \sum_{i=1}^{m} \underbrace{\log\left[1+\exp(-y_i\mathbf{w}^T\mathbf{x})\right]}_{loss\ \ function} + \underbrace{\frac{1}{\sigma^2}\mathbf{w}^T\mathbf{w}}_{regularizer} \tag{178}$$

31. Derive the EM algorithm for a Naïve Bayes classifier that is a mixture of Bernoulli distributions.

- **Expectation Maximization**:
  - A *meta-algorithm* to estimate a probability distribution when attributes are missing
  - Needs assumptions about the underlying probability distribution

    - Suited for generative models
    - Performance sensitive to the validity of the assumption (and also the initial guess of the parameters)
  - Converges to a local maximum of the likelihood function

Maximum Likelihood Estimation:

- Find parameters that maximize the likelihood (or log-likelihood) of the data

$$LL(data|parameters) = \sum_i \log\left[P(example_i|parameters)\right]$$

- Want to *maximize* $LL(data|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is a vector of our parameters. Start with

$$LL(data|\boldsymbol{\theta}) = \sum_i \log\left[\sum_y P(\mathbf{x}_i, y|\boldsymbol{\theta})\right] \tag{179}$$

We can introduce our probability distribution as $Q_i(y)$, resulting in

$$LL(data|\boldsymbol{\theta}) = \sum_i \log\left[\sum_y \left(Q_i(y) \cdot \frac{P(\mathbf{x}_i, y|\boldsymbol{\theta})}{Q_i(y)}\right)\right] \tag{180}$$

where

$$E_{z\sim Q}[f(z)] = \sum_z Q(z)f(z) \tag{181}$$

using this reduces our $LL(data|\boldsymbol{\theta})$ to

$$LL(data|\boldsymbol{\theta}) = \sum_i \log\left[E_{y\sim Q}\left(\frac{P(\mathbf{x}_i, y|\boldsymbol{\theta})}{Q_i(y)}\right)\right] \tag{182}$$

where we can simplify with *Jensen's Inequality* which states that for convex functions $f(E[x]) \leq E[f(x)]$ and for concave functions, $f(E[x]) \geq E[f(x)]$. Since our function is concave, we can say

$$\log\left[E_{y\sim Q_i}\left[\frac{P(\mathbf{x}_i, y|\boldsymbol{\theta})}{Q_i(y)}\right]\right] \geq E_{y\sim Q_i}\left[\log\left[\frac{P(\mathbf{x}_i, y|\boldsymbol{\theta})}{Q_i(y)}\right]\right] \tag{183}$$

which turns Equation (182) into

$$LL(data|\boldsymbol{\theta}) \geq \sum_i E_{y\sim Q_i}\left[\log\left[\frac{P(\mathbf{x}_i, y|\boldsymbol{\theta})}{Q_i(y)}\right]\right] \tag{184}$$

$$\geq \sum_i E_{y\sim Q_i}\left[\log\left[P(\mathbf{x}_i, y|\boldsymbol{\theta})\right]\right] - \sum_i E_{y\sim Q_i}\left[\log\left[Q_i(y)\right]\right] \tag{185}$$

Resuling in the final log-likelihood function as being

$$\mathcal{L}(\boldsymbol{\theta}; Q) = \sum_i E_{y\sim Q_i}\left[\log\left[P(\mathbf{x}_i, y|\boldsymbol{\theta})\right]\right] - \sum_i E_{y\sim Q_i}\left[\log\left[Q_i(y)\right]\right] \tag{186}$$

where when we're maximing this function, we can ignore the second term since it's independent of $\boldsymbol{\theta}$

Algorithm 3 ExpectationMaximization

Initialize the parameters $\boldsymbol{\theta}_{(0)}$

repeat

E-Step: For every example $\mathbf{x}_i$, estimate for every y

$$Q_i^{(t)} = P(y|\mathbf{x}_i, \boldsymbol{\theta}_{(t)})$$

M-Step: Find $\boldsymbol{\theta}_{(t+1)}$ by maximizing with respect to $\boldsymbol{\theta}$

$$\boldsymbol{\theta}_{(t+1)} = \max_{\boldsymbol{\theta}} \sum_i E_{y\sim Q_i}\left[\log\left[P(\mathbf{x}_i, y|\boldsymbol{\theta})\right]\right]$$

until convergence $(t = 1, 2, \ldots)$

return final $\boldsymbol{\theta}$

- **About the EM Algorithm**:
  - Will converge to a local maximum of the log-likelihood
    - Different initializations can give us different final estimates of probabilities
  - How many iterations
    - Till convergence. Keep track of the expected log-likelihood across iterations and if the change is smaller than some ϵ, then stop
  - What we need to specify for the learning algorithm
    - A task-specific definition of the probabilities
    - A way to solve the maximization (M step)
- **Expectation Maximization for Naïve Bayes**:
  - The setting:
    - Input: Features $\mathbf{x} \in \{0,1\}^d$
    - Output: $y \in \{0,1\}$
    - Dataset: $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_m\}$
  - Model:
    - $P(\mathbf{x}, y) = P(y) \prod_j P(x_j|y)$

Since it is a Bernoulli Distribution, we can define our *prior* and *likelihood* as

$$\textbf{Prior}: \tag{187}$$
$$P(y=1) = p \tag{188}$$
$$P(y=0) = (1-p) \tag{189}$$
$$\textbf{Likelihood}: \tag{190}$$
$$P(x_j = 1|y=1) = a_j \quad P(x_j = 0|y=1) = 1 - a_j \tag{191}$$
$$P(x_j = 1|y=0) = b_j \quad P(x_j = 0|y=0) = 1 - b_j \tag{192}$$

since we know that our data is independent and uniformly distributed, we can use the Bayes Assumption, where

$$\boldsymbol{\theta} = (p, a_1, a_2, \ldots, a_d, b_1, b_2, \ldots, b_d)^T \tag{193}$$
$$P(\mathbf{x}, y|\boldsymbol{\theta}) = P(y|\boldsymbol{\theta}) \prod_j P(x_j|y, \boldsymbol{\theta}) \tag{194}$$

E-step:

Goal: Suppose we have a current estimate of $\boldsymbol{\theta}$, compute $Q_i(y) = P(y|\mathbf{x}_i, \boldsymbol{\theta})$ for each example

$$P(y=1|\mathbf{x}_i, \boldsymbol{\theta}) = \frac{P(y=1, \mathbf{x}_i|\boldsymbol{\theta})}{P(y=1, \mathbf{x}_i|\boldsymbol{\theta}) + P(y=0, \mathbf{x}_i|\boldsymbol{\theta})} \tag{195}$$

which we can compute with our model of

$$P(\mathbf{x}, y|\boldsymbol{\theta}) = P(y|\boldsymbol{\theta}) \prod_j P(x_j|y, \boldsymbol{\theta}) \tag{196}$$

M-step:

Goal:

$$\boldsymbol{\theta}_{(t+1)} = \max_{\boldsymbol{\theta}} \sum_i E_{y \sim Q_i^{(t)}} \left[\log \left[P(\mathbf{x}_i, y|\boldsymbol{\theta})\right]\right] \tag{197}$$

Steps:

- Expand $log\left[P(\mathbf{x}_i, y|\boldsymbol{\theta})\right]$ in terms of p, a's, and b's

* Our original equation is defined by:

$$\boldsymbol{\theta}_{(t+1)} = \max_{\boldsymbol{\theta}} \sum_i E_{y \sim Q_i^{(t)}} \left[\log \left[P(\mathbf{x}_i, y | \boldsymbol{\theta}) \right] \right] \tag{198}$$

By using the model in Equation (196), we can expand the logarithm term. First, we need to calculate our prior and likelihood

$$P(y|\boldsymbol{\theta}) = p^{[y=1]}(1-p)^{[y=0]} \tag{199}$$

$$P(\mathbf{x}_i, y|\boldsymbol{\theta}) = a_j^{[x_j=1,y=1]}(1-a_j)^{[x_j=0,y=1]} \times b_j^{[x_j=1,y=0]}(1-b_j)^{[x_j=0,y=0]} \tag{200}$$

Throughout the rest of this proof, we are going to not write down the indicator functions, as they will be implied. It is easy to tell that we can't multiply any of the terms in $P(\mathbf{x}_i, y|\boldsymbol{\theta})$ together, and neither can we in $P(y|\boldsymbol{\theta})$, but we can cross multiply terms between the two. It is also easy to tell that any term with p corresponds to $y = 1$, any term with $(1 - a_j)$ corresponds to $[x_j = 0, y = 1]$, etc. Using this shorter notation, we can expand the logarithm term into

$$\log \left[P(\mathbf{x}_i, y|\boldsymbol{\theta}) \right] = \log(p) + \log(1-p) + \sum_j \log(a_j) + \log(1-a_j) + \log(b_j) + \log(1-b_j) \tag{201}$$

until the logarithm term is needed in expanded form, throughout the rest of this proof we will use the notation

$$\log[\alpha_j] = \log(p) + \log(1-p) + \sum_j \log(a_j) + \log(1-a_j) + \log(b_j) + \log(1-b_j) \tag{202}$$

– Substitute in Q_i to write down the full expectation

* The definition of $Q_i^{(t)}(y)$ is as follows:

$$Q_i^{(t)}(y) = P(y|\mathbf{x}_i, \boldsymbol{\theta}_{(t)}) \tag{203}$$

$$Q_i^{(t)}(y) = p^{[y=1]}(1-p)^{[y=0]} \tag{204}$$

we also know

$$E_{z \sim Q}[f(z)] = \sum_z f(z) Q(z) \tag{205}$$

where for the first part of the simplification we can use Equation (205) on Equation (198), where the logarithm term was expanded in the previous part. This result is

$$\boldsymbol{\theta}_{(t+1)} = \max_{\boldsymbol{\theta}} \sum_i \sum_y P(y|\mathbf{x}_i, \boldsymbol{\theta}_{(t)}) \log[\alpha_j] \tag{206}$$

where we can now plug in the defintion of $P(y|\mathbf{x}_i, \boldsymbol{\theta}_{(t)})$, giving us

$$\boldsymbol{\theta}_{(t+1)} = \max_{\boldsymbol{\theta}} \sum_i \sum_y p(1-p) \log[\alpha_j] \tag{207}$$

where we neglected the indicator functions again as they're implied. Now we can expand $\log[\alpha_j]$ term

$$\boldsymbol{\theta}_{(t+1)} = \max_{\boldsymbol{\theta}} \sum_i \sum_y p(1-p) \left(\log(p) + \log(1-p) + \sum_j \log(a_j) + \log(1-a_j) + \cdots \right.$$
$$\left. \cdots + \log(b_j) + \log(1-b_j) \right) \tag{208}$$

where we can multiply through Equation (208) efficiently. In other words, we ignore terms such that the indicator variables may differ, as either of the terms will be zero all the time. For example, p has the indicator function $[y = 1]$ and $\log(b_j)$ has the indicator function $[x_j = 1, y = 0]$, where as you can see they differ for the y's. In this instance, we would *only* multiply $\log(b_j)$ by $(1 - p)$ since they have compatable indicator functions. Doing this, we get the result

$$\begin{aligned}\boldsymbol{\theta}_{(t+1)} = \max_{\boldsymbol{\theta}} \sum_i \sum_y p\log(p) + (1-p)\log(1-p) + \cdots \\ \cdots + \sum_j p\log(a_j) + p\log(1-a_j) + (1-p)\log(b_j) + (1-p)\log(1-b_j)\end{aligned} \tag{209}$$

- Take the derivative with respect to each p, a_j, and b_j
  - The equation that we want to maximize is Equation (209), which we can do so by taking the gradient with respect to all the variabes, and setting them equal to zero. Therefore, we can remove the $\max_{\boldsymbol{\theta}}$ term, since that's what we're doing in this process.

$$\frac{\partial \boldsymbol{\theta}_{(t+1)}}{\partial p} = (1 + \log(p)) + (1 - \log(1-p)) + \sum_j \log(a_j) + \log(1-a_j) - \log(b_j) - \log(1-b_j) \tag{210}$$

where we can simplify this by only looking at the terms that correspond to $y = 1$. We can do this to make the problem we're trying to solve either, and since $P(y = 1|\boldsymbol{\theta}) = p$, it makes intuitive sense to restrict our space to the case $y = 1$, since $P(y = 0|\boldsymbol{\theta}) = (1 - p)$. In doing so, we get

$$\frac{\partial \boldsymbol{\theta}_{(t+1)}}{\partial p} = \sum_i \sum_y 1 + \log(p) + \sum_j \log(a_j) + \log(1-a_j) \Bigg|_{[y=1]} \tag{211}$$

$$\frac{\partial \boldsymbol{\theta}_{(t+1)}}{\partial a_j} = \sum_i \sum_y \sum_j \frac{p}{a_j} + \frac{p}{1-a_j} \tag{212}$$

similiarly, for b_j

$$\frac{\partial \boldsymbol{\theta}_{(t+1)}}{\partial b_j} = -\sum_i \sum_y \sum_j \frac{p}{b_j} + \frac{p}{1-b_j} \tag{213}$$

- Set derivatives to zero to get a new estimate for p, a_j, and b_j
  - Finally, we can take Equations (211, 212, 213) and set them equal to zero to get how to maximize them. In doing so

***p* :**

$$0 = \sum_i \sum_y 1 + \log(p) + \sum_j \log(a_j) + \log(1-a_j) \tag{214}$$

we can raise this to the exponent, giving

$$e^0 = \exp\left(\sum_i \sum_y 1 + \log(p) + \sum_j \log(a_j) + \log(1-a_j)\right) \tag{215}$$

$$1 = \exp\left(\sum_y 1 + \log(p)\right) + \exp\left(\sum_i \sum_y \sum_j \underbrace{\log(a_j) + \log(1-a_j)}_{\alpha_j}\right) \tag{216}$$

which simplifies to

$$1 = \exp\left(\sum_y 1\right) \cdot \sum_y p + \sum_i \sum_y \sum_j \alpha_j \tag{217}$$

$$1 = e^m \sum_y p + \sum_i \sum_y \sum_j \alpha_j \tag{218}$$

where we can set e^m to 1 since we're trying to maximize over a given set of parameters which are independent of that term. We can solve for p, giving us

$$\sum_y p = -\sum_i \sum_y \sum_j \alpha_j \tag{219}$$

$$p = \frac{-\sum_i \sum_y \sum_j \alpha_j}{\sum_y} \tag{220}$$

where we can rewrite it as positive, because since it's a probability, it is strictly > 0

$$p = \frac{\sum_i \sum_y \sum_j \alpha_j}{\sum_y} \tag{221}$$

Where the numerator is all terms that correspond to $y = 1$ and the denominator is the total number of y

a_j :

$$0 = \sum_i \sum_y \sum_j \frac{p}{a_j} + \frac{p}{1 - a_j} \tag{222}$$

$$\sum_i \sum_y \sum_j \frac{p}{a_j} = -\sum_i \sum_y \sum_j \frac{p}{1 - a_j} \tag{223}$$

$$\sum_i \sum_y \sum_j \frac{p(1 - a_j)}{a_j} = -\sum_i \sum_y \sum_j p \tag{224}$$

where p is independent of all the sums, so we can multiply p by the largest term of each sum, resulting in $p' = m^2 d\cdot$. We can divide by p on the left hand side, giving us

$$\sum_i \sum_y \sum_j \frac{1 - a_j}{a_j} = -\frac{p'}{\sum_i \sum_y \sum_j p} \tag{225}$$

since this is a maximization problem, we can take the logarithm of both sides, giving us

$$\log \left[\sum_i \sum_y \sum_j \frac{1 - a_j}{a_j} \right] = \log \left[-\frac{p'}{\sum_i \sum_y \sum_j p} \right] \tag{226}$$

$$\log \left[\sum_i \sum_y \sum_j 1 - a_j \right] - \log \left[\sum_i \sum_y \sum_j a_j \right] = \log \left[-p' \right] - \log \left[\sum_i \sum_y \sum_j p \right] \tag{227}$$

taking an exponential of both sides to get rid of the logarithms

$$\sum_i \sum_y \sum_j 1 - a_j - \sum_i \sum_y \sum_j a_j = -p' - \sum_i \sum_y \sum_j p \tag{228}$$

$$\sum_i \sum_y \sum_j 1 - 2a_j = -2p' \tag{229}$$

$$\sum_i \sum_y \sum_j 2a_j - 1 = 2p' \tag{230}$$

$$\sum_i \sum_y \sum_j 2a_j = 2p' + \sum_i \sum_y \sum_j 1 \tag{231}$$

$$\sum_i \sum_y \sum_j 2a_j = 2p \cdot m^2 d + m^2 d \tag{232}$$

where, since this is a maximization problem, we can get rid of all terms that are constants, as it will not contribute to maximizing the equation. This gives us

$$\sum_i \sum_y \sum_j a_j = p \tag{233}$$

where p was solved in the previous section as

$$\sum_i \sum_y \sum_j a_j = \frac{\sum_i \sum_j \alpha_j}{\sum_y} \tag{234}$$

but the term on the right is only for the instance $[x_{i,j} = 1, y = 1]$, so the summation on the right hand side for α_j would only go over those as well. The denominator would only sum over terms for $[y = 1]$ since that is what a_j is restricted to. This results in the nuerator representing the total number of terms such that $[x_{i,j} = 1, y = 1]$ and the denominator representing the total number of terms such that $[y = 1]$.

$\boldsymbol{b_j}$:

This is very similar to the previous section for a_j, as the differentials are almost exactly the same. Instead of redoing it, we can transform our maximiation problem for b_j into the same as that of a_j, by when you first set the equation equal to zero, you multiply through by -1, which will give you the equation of the exact same form. Exploiting this, we don't need to redo this derivation for b_j and can simply use the same logic at the end to write down the answer since the form compared to a_j is the same. This gives us the following, where β_j is analagous to the α_j term given in the previous problem

$$\sum_i \sum_y \sum_j b_j = p = \frac{\sum_i \sum_y \sum_j \beta_j}{\sum_y} \tag{235}$$

- The final equations for $p,\ a_j,\ b_j$ are written as

$$p = \frac{\text{SoftCount}\,(y = 1)}{\text{SoftCount}\,(y = 1) + \text{SoftCount}\,(y = 0)} \tag{236}$$

$$a_j = \frac{\text{SoftCount}\,(y = 1, x_j = 1)}{\text{SoftCount}\,(y = 1)} \tag{237}$$

$$b_j = \frac{\text{SoftCount}\,(y = 0, x_j = 1)}{\text{SoftCount}\,(y = 0)} \tag{238}$$

where

$$\text{SoftCount}\,(y = 1) = \sum_i P(y = 1|\mathbf{x}_i, \boldsymbol{\theta}_{(t)}) \tag{239}$$

$$\text{SoftCount}\,(y = 1, x_j = 1) = \sum_i P(y = 1|\mathbf{x}_i, \boldsymbol{\theta}_{(t)})[x_{i,j} = 1] \tag{240}$$

$$\text{SoftCount}\,(y = 0, x_j = 1) = \sum_i P(y = 0|\mathbf{x}_i, \boldsymbol{\theta}_{(t)})[x_{i,j} = 1] \tag{241}$$