Homework 3 - Solutions

1. **[7pts] AdaBoost.**

   (a) **[5pts]** The goal of this question is to show that the AdaBoost algorithm changes the weights in order to force the weak learner to focus on difficult data points. Here we consider the case that the target labels are from the set $\{-1, +1\}$ and the weak learner also returns a classifier whose outputs belongs to $\{-1, +1\}$ (instead of $\{0, 1\}$). Consider the t-th iteration of AdaBoost, where the weak learner is

$$h_t \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i=1}^{N} w_i \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t^{(i)}\},$$

   the w-weighted classification error is

$$\text{err}_t = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i},$$

   and the classifier coefficient is $\alpha_t = \frac{1}{2} \log \frac{1-\text{err}_t}{\text{err}_t}$. (Here, log denotes the natural logarithm.) AdaBoost changes the weights of each sample depending on whether the weak learner h_t classifies it correctly or incorrectly. The updated weights for sample i is denoted by w_i' and is

$$w_i' \leftarrow w_i \exp\Big(-\alpha_t t^{(i)} h_t(\mathbf{x}^{(i)})\Big).$$

   Show that the error w.r.t. $(w_1', \ldots, w_N')$ is exactly $\frac{1}{2}$. That is, show that

$$\text{err}_t' = \frac{\sum_{i=1}^{N} w_i' \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i'} = \frac{1}{2}.$$

   Note that here we use the weak learner of iteration t and evaluate it according to the new weights, which will be used to learn the $t+1$-st weak learner. What is the interpretation of this result?

   Hints:

   i. Start from err_t' and divide the summation to two sets of $E = \{i : h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}$ and its complement $E^c = \{i : h_t(\mathbf{x}^{(i)}) = t^{(i)}\}$.

   ii. Note that

$$\frac{\sum_{i \in E} w_i}{\sum_{i=1}^{N} w_i} = \text{err}_t.$$

Solution.

$$
\begin{aligned}
\text{err}'_t &= \frac{\sum_{i=1}^N w'_i \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^N w'_i} \\
&= \frac{\sum_{i\in E} w_i \exp(\alpha_t)}{\sum_{i\in E} w_i \exp(\alpha_t) + \sum_{i\in E^C} w_i \exp(-\alpha_t)} && \triangleright \text{ use hint (a), expand } w'_i \\
&= \frac{\sum_{i\in E} w_i}{\sum_{i\in E} w_i + \exp(-2\alpha_t)\sum_{i\in E^C} w_i} && \triangleright \text{ divide by } \exp(\alpha_t)/\exp(\alpha_t) \\
&= \frac{\sum_{i\in E} w_i / \sum_{i=1}^N w_i}{\left(\sum_{i\in E} w_i + \exp(-2\alpha_t)\sum_{i\in E^C} w_i\right)/\sum_{i=1}^N w_i} && \triangleright \text{ divide by } \Sigma w_i/\Sigma w_i \\
&= \frac{\text{err}_t}{\text{err}_t + \frac{\text{err}_t}{1-\text{err}_t}(1-\text{err}_t)} && \triangleright \text{ use hint (b)} \\
&= \frac{1}{2}
\end{aligned}
$$

Since the t-th weak learner was assumed to have $\text{err}_t < \frac{1}{2}$, its error has increased under the new weights. This means that relatively more weight (or focus) has been given to data points on which it made mistakes.

(b) **[2pts]** Recall from lecture that we can rewrite the 0-1 loss as: $\mathbb{I}[h(x^{(n)}) \neq t^{(n)}] = \frac{1}{2}(1 - h(x^{(n)}) \cdot t^{(n)})$. Use this identity to prove that the weight update from part (a):

$$w'_i \leftarrow w_i \exp\Big(-\alpha_t t^{(i)} h_t(\mathbf{x}^{(i)})\Big).$$

is proportional, up to a constant factor, to weight update:

$$w'_i \leftarrow w_i \exp\Big(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}\Big).$$

What is the constant factor? Since we normalize the weights when computing err_t, these two updates are equivalent.

Solution.

$$
\begin{aligned}
w_i \exp\Big(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}\Big) &= w_i \exp\Big(\alpha_t\Big(1 - h(x^{(n)}) \cdot t^{(n)}\Big)\Big) && \triangleright \text{ use identity} \\
&= w_i \exp(\alpha_t) \exp\Big(-\alpha_t t^{(i)} h_t(\mathbf{x}^{(i)})\Big) \\
&\propto w_i \exp\Big(-\alpha_t t^{(i)} h_t(\mathbf{x}^{(i)})\Big)
\end{aligned}
$$

where the constant factor is $\exp(\alpha_t)$.

2. **[13pts] Fitting a Naïve Bayes Model**

In this question, we'll fit a Naïve Bayes model to the MNIST digits using maximum likelihood. The starter code will download the dataset and parse it for you: Each training sample $(\mathbf{t}^{(i)}, \mathbf{x}^{(i)})$ is composed of a vectorized binary image $\mathbf{x}^{(i)} \in \{0,1\}^{784}$, and 1-of-10 encoded class label $\mathbf{t}^{(i)}$, i.e., $t_c^{(i)} = 1$ means image i belongs to class c.

Given parameters $\boldsymbol{\pi}$ and $\boldsymbol{\theta}$, Naïve Bayes defines the joint probability of the each data point $\mathbf{x}$ and its class label c as follows:

$$p(\mathbf{x}, c \,|\, \boldsymbol{\theta}, \boldsymbol{\pi}) = p(c \,|\, \boldsymbol{\theta}, \boldsymbol{\pi}) p(\mathbf{x} \,|\, c, \boldsymbol{\theta}, \boldsymbol{\pi}) = p(c \,|\, \boldsymbol{\pi}) \prod_{j=1}^{784} p(x_j \,|\, c, \theta_{jc}).$$

where $p(c \,|\, \boldsymbol{\pi}) = \pi_c$ and $p(x_j = 1 \,|\, c, \boldsymbol{\theta}, \boldsymbol{\pi}) = \theta_{jc}$. Here, $\boldsymbol{\theta}$ is a matrix of probabilities for each pixel and each class, so its dimensions are 784×10, and $\boldsymbol{\pi}$ is a vector with one entry for each class. (Note that in the lecture, we simplified notation and didn't write the probabilities conditioned on the parameters, i.e. $p(c|\boldsymbol{\pi})$ is written as $p(c)$ in lecture slides).

For binary data ($x_j \in \{0,1\}$), we can write the Bernoulli likelihood as

$$p(x_j \,|\, c, \theta_{jc}) = \theta_{jc}^{x_j} (1 - \theta_{jc})^{(1-x_j)}, \tag{0.1}$$

which is just a way of expressing $p(x_j = 1|c, \theta_{jc}) = \theta_{jc}$ and $p(x_j = 0|c, \theta_{jc}) = 1 - \theta_{jc}$ in a compact form. For the prior $p(\mathbf{t} \,|\, \boldsymbol{\pi})$, we use a categorical distribution (generalization of Bernoulli distribution to multi-class case),

$$p(t_c = 1 \,|\, \boldsymbol{\pi}) = p(c \,|\, \boldsymbol{\pi}) = \pi_c \;\text{ or equivalently }\; p(\mathbf{t} \,|\, \boldsymbol{\pi}) = \Pi_{j=0}^{9} \pi_j^{t_j} \quad \text{where} \quad \sum_{i=0}^{9} \pi_i = 1,$$

where $p(c \,|\, \boldsymbol{\pi})$ and $p(\mathbf{t} \,|\, \boldsymbol{\pi})$ can be used interchangeably. You will fit the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$ using MLE and MAP techniques. In both cases, your fitting procedure can be written as a few simple matrix multiplication operations.

(a) **[3pts]** First, derive the *maximum likelihood estimator* (MLE) for the class-conditional pixel probabilities $\boldsymbol{\theta}$ and the prior $\boldsymbol{\pi}$. Derivations should be rigorous.

Hint 1: We saw in lecture that MLE can be thought of as 'ratio of counts' for the data, so what should $\hat{\theta}_{jc}$ be counting?

Hint 2: Similar to the binary case, when calculating the MLE for π_j for $j = 0, 1, ..., 8$, write $p(\boldsymbol{t}^{(i)} \,|\, \boldsymbol{\pi}) = \Pi_{j=0}^{9} \pi_j^{t_j^{(i)}}$ and in the log-likelihood replace $\pi_9 = 1 - \Sigma_{j=0}^{8} \pi_j$, and then take derivatives w.r.t. π_j. This will give you the ratio $\hat{\pi}_j / \hat{\pi}_9$ for $j = 0, 1, .., 8$. You know that $\hat{\pi}_j$'s sum up to 1.

Solution. We observe m samples (images), $(\mathbf{t}^{(n)}, \mathbf{x}^{(n)})$ for $n = 1, ..., m$. We can represent the likelihood is:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\pi} | \{\mathbf{t}^{(n)}, \mathbf{x}^{(n)}\}_{n=1}^{m}) = \prod_{n=1}^{m} \prod_{i=0}^{9} \pi_i^{t_i^{(n)}} \prod_{j=1}^{784} \theta_{ji}^{x_j^{(n)}} (1 - \theta_{ji})^{1 - x_j^{(n)}}$$

Taking a logarithm to examine log-likelihood, we get:

$$\ell(\boldsymbol{\theta}, \boldsymbol{\pi} | \{\mathbf{t}^{(n)}, \mathbf{x}^{(n)}\}_{n=1}^{m}) = \sum_{n=1}^{m} \sum_{i=0}^{9} \left[t_i^{(n)} \log \pi_i + \left(\sum_{j=1}^{784} x_j^{(n)} \log \theta_{ji} + (1 - x_j^{(n)}) \log(1 - \theta_{ji}) \right) \right]$$

To obtain the maximum likelihood estimate of $\boldsymbol{\theta}$, we take a derivative with respect to θ_{ji} and set it equal to 0.

$$\begin{aligned}
\frac{d\ell}{d\theta_{ji}} = 0 &\Leftrightarrow \sum_{n=1}^{m} t_i^{(n)} \left(\frac{x_j^{(n)}}{\theta_{ji}} - \frac{1 - x_j^{(n)}}{1 - \theta_{ji}} \right) = 0 \\
&\Leftrightarrow \sum_{n=1}^{m} t_i^{(n)} \Big(x_j^{(n)}(1 - \theta_{ji}) - \theta_{ji}(1 - x_j^{(n)}) \Big) = 0 \\
&\Leftrightarrow \sum_{n=1}^{m} t_i^{(n)} \Big(x_j^{(n)} - \theta_{ji} \Big) = 0 \\
&\Leftrightarrow \hat{\theta}_{ji} = \frac{\sum_{n=1}^{m} t_i^{(n)} x_j^{(n)}}{\sum_{n=1}^{m} t_i^{(n)}}
\end{aligned}$$

To obtain the maximum likelihood estimate of $\boldsymbol{\pi}$, we follow the hint, then take a derivative with respect to π_i for $i = 0 \dots 8$ and set it equal to 0. This works because even though the resulting optimization problem is still constrained (we are optimizing over the set $\{(\pi_i)_{i=0\dots8} | \pi_i \geq 0, \sum_i \pi_i \leq 1\}$), the likelihood is a concave function that approaches $-\infty$ as we approach the boundary of this new constraint set. Thus, the function attains its maximum in the interior of the new constraint set. We also accepted solutions using Lagrange multipliers to directly solve the original constrained optimization problem.

$$\begin{aligned}
\ell(\boldsymbol{\theta}, \boldsymbol{\pi} | \{\mathbf{t}^{(n)}, \mathbf{x}^{(n)}\}_{n=1}^{m}) &= \sum_{n=1}^{m} \sum_{i=0}^{9} \left[t_i^{(n)} \log \pi_i + \left(\sum_{j=1}^{784} x_j^{(n)} \log \theta_{ji} + (1 - x_j^{(n)}) \log(1 - \theta_{ji}) \right) \right] \\
&= \sum_{n=1}^{m} \Big[\Sigma_{j=0}^{8} t_j^{(n)} \log \pi_j + t_9^{(n)} \log(1 - \Sigma_{j=0}^{8} \pi_j) + (\dots) \Big]
\end{aligned}$$

$$\begin{aligned}
\frac{d\ell}{d\pi_i} = 0 &\Leftrightarrow \sum_{n=1}^{m} \left[\Sigma_{j=0}^{8} \frac{t_j^{(n)}}{\pi_j} + \frac{t_9^{(n)}}{1 - \Sigma_{j=0}^{8} \pi_j} \right] = 0 \\
&\Leftrightarrow \sum_{n=1}^{m} \left[\frac{t_i^{(n)}}{\pi_i} + \frac{t_9^{(n)}}{1 - \Sigma_{j=0}^{8} \pi_j} \right] = 0 \\
&\Leftrightarrow \sum_{n=1}^{m} \Big[t_i^{(n)} \pi_9 + t_9^{(n)} \pi_i \Big] = 0 \\
&\Leftrightarrow \frac{\hat{\pi}_i}{\hat{\pi}_9} = \frac{\sum_{n=1}^{m} t_i^{(n)}}{\sum_{n=1}^{m} t_9^{(n)}}.
\end{aligned}$$

Now using $\sum_j \hat{\pi}_j = 1$ we get:

$$\frac{1}{\hat{\pi}_9} = \frac{m}{\sum_{n=1}^{m} t_9^{(n)}}$$

so that (using the same argument for each j), we have $\hat{\pi}_j = \sum_n t_j^{(n)} / m$.

(b) **[1pt]** Derive the log-likelihood $\log p(\mathbf{t}|\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\pi})$ for a single training image.

Solution. Observe that:

$$P(\mathbf{t}|\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\pi}) = \frac{P(\mathbf{x}, \mathbf{t}|\boldsymbol{\theta}, \boldsymbol{\pi})}{P(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\pi})}$$

Taking a logarithm to $p(c|\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\pi})$ to obtain the log-likelihood, we get:

$$\begin{aligned}
\log p(\mathbf{t}|\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\pi}) &= \log p(\mathbf{x}, \mathbf{t}|\boldsymbol{\theta}, \boldsymbol{\pi}) - \log p(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\pi}) \\
&= \log p(\mathbf{t}|\boldsymbol{\pi}) + \sum_{j=1}^{784} \log p(x_j|\mathbf{t}, \boldsymbol{\theta}) - \log p(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\pi}) \\
&= \log p(\mathbf{t}|\boldsymbol{\pi}) + \sum_{j=1}^{784} \log p(x_j|\mathbf{t}, \boldsymbol{\theta}) - \log \left(\sum_{i=0}^{9} p(t_i = 1|\boldsymbol{\pi}) \prod_{j=1}^{784} p(x_j|t_i = 1, \boldsymbol{\theta}, \boldsymbol{\pi}) \right)
\end{aligned}$$

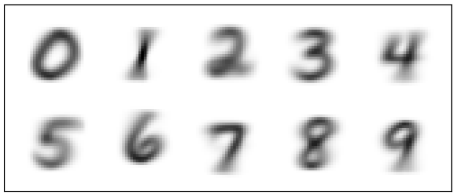
Note that the last constant, we have to marginalize over the class.

(c) **[3pt]** Fit the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$ using the training set with MLE, and try to report the average log-likelihood per data point $\frac{1}{N}\Sigma_{i=1}^{N} \log p(\mathbf{t}^{(i)}|\mathbf{x}^{(i)}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}})$, using Equation (0.1). What goes wrong? (it's okay if you can't compute the average log-likelihood here).

Solution. `Average log-likelihood for MLE is nan`. One of the pixels is off/0 (or alternatively on/1) for the entire dataset, so we end up taking log of 0. (Note: this can work on the training set if you implement the log-likelihood to account for the edge case, e.g., define $0^0 = 1$, (not necessary), but it can have problems on the test set).

(d) **[1pt]** Plot the MLE estimator $\hat{\boldsymbol{\theta}}$ as 10 separate greyscale images, one for each class.

Solution.



(e) **[2pt]** Derive the *Maximum A posteriori Probability* (MAP) estimator for the class-conditional pixel probabilities $\boldsymbol{\theta}$, using a Beta(3, 3) prior on each θ_{jc}. Hint: it has a simple final form, and you can ignore the Beta normalizing constant.

Solution. Our goal is to maximize the posterior, i.e:

$$\begin{aligned}\arg\max_\theta p(\theta|\mathcal{D}) &= \arg\max_\theta p(\mathcal{D}|\theta)p(\theta) \\ &= \arg\max_\theta [\log p(\mathcal{D}|\theta) + \log p(\theta)].\end{aligned}$$

In our particular case, we have a Beta prior on each θ_{jc}.

$$p(\theta_{jc}|\alpha,\beta) = \frac{\theta_{jc}^{\alpha-1} + (1-\theta_{jc})^{\beta-1}}{\mathrm{B}(\alpha,\beta)}$$

where $\mathrm{B}(\alpha,\beta)$ is a normalization constant. Taking a logarithm:

$$\log p(\theta_{jc}|\alpha,\beta) = (\alpha-1)\log\theta_{jc} + (\beta-1)\log(1-\theta_{jc}) - \log \mathrm{B}(\alpha,\beta)$$

In the previous question, we obtained the MLE of θ_{jc}. Using the equations obtained from the previous question, we get:

$$\begin{aligned}\frac{d}{d\theta_{jc}}\log p(\theta|\mathcal{D}) = 0 &\Leftrightarrow \sum_{n=1}^m t_c^{(n)}\left(\frac{x_j^{(n)}}{\theta_{jc}} - \frac{1-x_j^{(n)}}{1-\theta_{jc}}\right) + \frac{\alpha-1}{\theta_{jc}} - \frac{\beta-1}{1-\theta_{jc}} = 0 \\ &\Leftrightarrow \hat{\theta}_{jc} = \frac{\alpha - 1 + \sum_{n=1}^m t_c^{(n)} x_j^{(n)}}{\alpha+\beta-2+\sum_{n=1}^m t_c^{(n)}}\end{aligned}$$

Assigning $\alpha = 3$ and $\beta = 3$, we obtain:

$$\hat{\theta}_{jc} = \frac{2 + \sum_{n=1}^m t_c^{(n)} x_j^{(n)}}{4+\sum_{n=1}^m t_c^{(n)}}$$

Compared to the MLE, the MAP solution adds 4 at the bottom and 2 at the top. This is beneficial when some examples were missing from the training set.

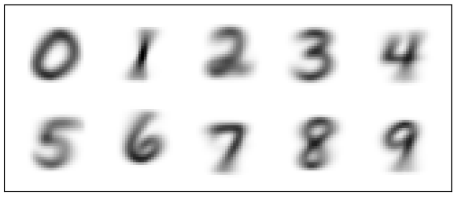
(f) **[2pt]** Fit the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$ using the training set with MAP estimators from previous part, and report both the average log-likelihood per data point, $\frac{1}{N}\Sigma_{i=1}^N \log p(\mathbf{t}^{(i)}|\mathbf{x}^{(i)}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}})$, and the accuracy on both the training and test set. The accuracy is defined as the fraction of examples where the true class is correctly predicted using $\hat{c} = \mathrm{argmax}_c \log p(t_c = 1|\mathbf{x}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}})$.

Solution.

```
Average log-likelihood for MAP is -3.3570631378602847
Training accuracy for MAP is 0.8352166666666667
Test accuracy for MAP is 0.816
```

(g) **[1pt]** Plot the MAP estimator $\hat{\boldsymbol{\theta}}$ as 10 separate greyscale images, one for each class.

Solution.



3. **[4pts] Generating from a Naïve Bayes Model**

Defining a joint probability distribution over the data lets us generate new data, and also lets us answer all sorts of queries about the data. This is why these models are called *Generative Models*. We will use the Naïve Bayes model trained in previous question to generate data.

(a) **[1pt]** True or false: Given this model's assumptions, any two pixels x_i and x_j where $i \neq j$ are independent given c.

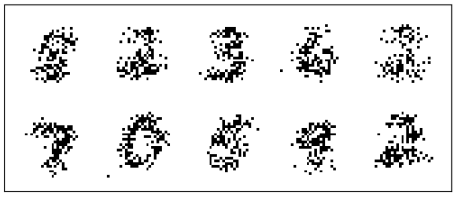
Solution. True. This follows directly from the Naive Bayes assumption.

(b) **[1pt]** True or false: Given this model's assumptions, any two pixels x_i and x_j where $i \neq j$ are independent after marginalizing over c.

Solution. False. From NB, we only assume that the two pixels x_i and x_j where $i \neq j$ are only independent given c. One counter example for this question is the result from d.

(c) **[2pts]** Using the parameters fit using MAP in Question 1, produce random image samples from the model. That is, randomly sample and plot 10 binary images from the marginal distribution $p(\mathbf{x}|\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}})$. Hint: To sample from $p(\mathbf{x} \mid \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\pi}})$, first sample random variable c from $p(c \mid \hat{\boldsymbol{\pi}})$ using `np.random.choice`, then depending on the value of c, sample x_j from $p(x_j \mid c, \hat{\theta}_{jc})$ for $j = 1, ..., 784$ using `np.random.binomial(1,..)`. These functions can take matrix probabilities as input, so your solution to this part should be a few lines of code.

Solution.



(d) **[Optional]** One of the advantages of generative models is that they can handle missing data, or be used to answer different sorts of questions about the model. Derive $p(\mathbf{x}_{bottom}|\mathbf{x}_{top}, \boldsymbol{\theta}, \boldsymbol{\pi})$, the marginal distribution of a single pixel in the bottom half of an image given the top half, conditioned on your fit parameters. Hint: you'll have to marginalize over c.

(e) **[Optional]** For 20 images from the training set, plot the top half the image concatenated with the marginal distribution over each pixel in the bottom half. i.e. the bottom half of the image should use grayscale to represent the marginal probability of each pixel being 1 (darker for values close to 1).