Phileas Hocquard
phileas@cs.toronto.edu

Probabilistic Learning and Reasoning, #2

Problem 1 (Basic Naïve Bayes, 10 points)
In this question, we'll fit a naïve Bayes model to the MNIST digits using maximum likelihood. Naïve Bayes defines the joint probability of the each datapoint $\mathbf{x}$ and its class label c as follows:

$$p(\mathbf{x}, c|\boldsymbol{\theta}, \pi) = p(c|\pi)p(\mathbf{x}|c, \theta_c) = p(c|\pi)\prod_{d=1}^{784} p(x_d|c, \theta_{cd}) \tag{1}$$

For binary data, we can use the Bernoulli likelihood:

$$p(x_d|c, \theta_{cd}) = Ber(x_d|\theta_{cd}) = \theta_{cd}^{x_d}(1-\theta_{cd})^{(1-x_d)} \tag{2}$$

Which is just a way of expressing that $p(x_d = 1|c, \theta_{cd}) = \theta_{cd}$.
For $p(c|\pi)$, we can just use a categorical distribution:

$$p(c|\pi) = Cat(c|\pi) = \pi_c \tag{3}$$

Note that we need $\sum_{i=0}^{9} \pi_i = 1$.

(a) **Derive the *maximum likelihood estimate* (MLE) for the class-conditional pixel means $\boldsymbol{\theta}$. Hint: We saw in lecture that MLE can be thought of as 'counts' for the data, so what should $\hat{\theta}_{cd}$ be counting?**

N represents the size of the training elements where for each element with have pixel data and class that are observable.N_{cd} is the number of training elements from the class with a respective pixel d value of 1.

$$P(x, c|\pi, \theta) = \prod_c \pi_c^{N_c} \prod_n \prod_d \theta_{c_n d}^{x_{nd}}(1-\theta_{c_n d})^{1-x_{x_{nd}}}$$

$$P(x, c|\pi, \theta) \propto \prod_{c,d} (\theta_{cd})^{N_{cd}}(1-\theta_{cd})^{1-N_{cd}}$$

$$\log(P(\theta, x, c|\pi)) = \sum (N_{cd})\log\theta_{cd} + (1-N_{cd})\log(1-\theta_{cd})$$

$$\frac{\partial}{\partial\theta_{cd}}\log\prod_d \theta_{c_d}^{x_d}(1-\theta_{c_d})^{1-x_d} = \frac{\partial}{\partial\theta_{cd}}\sum_d (x_d)\log\theta_{cd} + (1-x_d)\log(1-\theta_{cd})$$

$$.\propto \frac{x_d}{\theta_{cd}}\frac{1-x_d}{1-\theta_{cd}}, \textit{when setting the derivative to.}\ 0:$$

$$\theta_{kd,mle} = \frac{\sum_e^N 1(c^{(e)}=k)x_d}{\sum_e^N 1(c^{(e)}=k)}$$

(b) **Derive the *maximum a posteriori* (MAP) estimate for the class-conditional pixel means $\boldsymbol{\theta}$, using a Beta(2, 2) prior on each θ. Hint: it has a simple final form, and you can ignore the Beta normalizing constant.**

Similar to previously N_c corresponds to the number of elements from class c and N_{cd} the number of elements from class c with pixel d equal to 1. The maximum a posteriori makes use of the θ prior.

$$P(\theta|x,c,\pi)=\frac{P(\theta,x,c|\pi)}{P(x,c|\pi)}$$
$$P(\theta|\pi)P(x,c|\theta,\pi)=P(\theta,x,c|\pi)$$

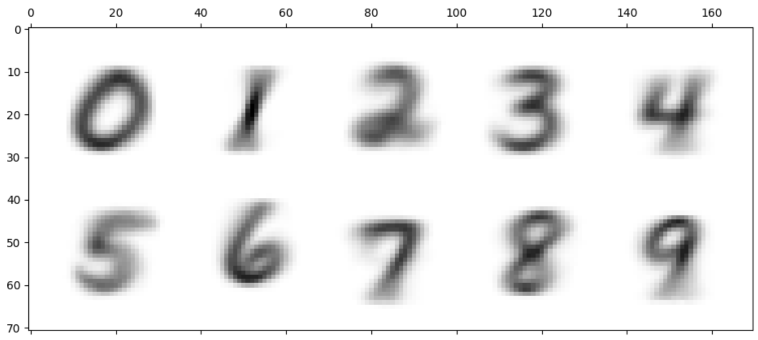
$$P(\theta,x,c|\pi)=\prod_c \pi_c^{N_c}\prod_n\prod_d \theta_{c_n d}^{x_{nd}}(1-\theta_{c_n d})^{1-x_{x_{nd}}}\prod_{c,d}\frac{(\theta_{cd})(1-\theta_{cd})}{\beta(4)}$$
$$P(\theta,x,c|\pi)\propto\prod_{c,d}(\theta_{cd})^{1+N_{cd}}(1-\theta_{cd})^{1+N_c-N_{cd}}$$
$$\log(P(\theta,x,c|\pi))=\sum_{c,d}(1+N_{cd})\log\theta_{cd}+(1+N_c-N_{cd})\log(1-\theta_{cd})$$

We now need to derive the log of our function, since to maximize the above we need to take the partial derivatives with respect to θ_{cd} and set to zero. We obtain the following doing so:

$$\frac{1+N_{cd}}{\theta_{cd}}-\frac{1+N_c-N_{cd}}{1-\theta_{cd}}=0$$

$$\theta_{cd,map}=\frac{1+N_{cd}}{2+N_c}\propto\theta_{kd,map}=\frac{1+\sum_e^N 1(c^{(e)}=k)x_d}{2+\sum_e^N 1(c^{(e)}=k)}$$

(c) **Fit θ to the training set using the MAP estimator. Plot θ as 10 separate greyscale images, one for each class.**



(d) **Derive the predictive log-likelihood $\log p(c|\mathbf{x},\boldsymbol{\theta},\pi)$ for a single training image.**

For a single training example we can obtain $\log p(c|\mathbf{x},\boldsymbol{\theta},\pi)$ by marginalizing over c on the denominator as follows:

$$p(c|\pi, x, \theta) = \frac{p(c, x|\pi, \theta)}{\sum_c p(c, x|\pi, \theta)}$$

$$p(c|\pi, x, \theta) = \frac{\frac{1}{10} + \sum_{d=1}^{784} x_d * \log(\theta_{cd}) + (1-x_d)\log(1-\theta_{cd})}{\sum_{k=0}^{9} \frac{1}{10} + \sum_{d=1}^{784} x_d * \log(\theta_{kd}) + (1-x_d)\log(1-\theta_{kd})}$$

Problem 2 (Advanced Naïve Bayes, 10 points)

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.

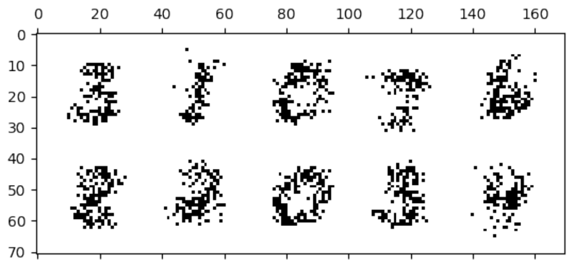
(a) **True or false: Given our model's assumptions, any two pixels x_i and x_j where $i \neq j$ are independent given c.**

True. The product of the probability distributions of the individuals pixels is equivalent to having the model given the class and the probability distributions of the pixels of the data.

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

This is False. Since for a given model the probablity of the class depends on the value of each individual pixel, and when given the class of the data the probability distribution of another pixel depends on it.

(c) **Using the parameters fit in question 1, produce random image samples from the model. That is, randomly sample and plot 10 binary images from the marginal distribution $p(\mathbf{x}|\boldsymbol{\theta}, \boldsymbol{\pi})$.**
To obtain each pixel value randomly we make use of the binomial function under a single dimension, as it is equivalent for 1 pixel.



(d) **Derive $p(\mathbf{x}_{bottom}|\mathbf{x}_{top}, \boldsymbol{\theta}, \pi)$, the joint distribution over the bottom half of an image given the top half, conditioned on your fit parameters.**

We define two different sets $top \wedge bottom$, *who hold* 392 *pixels each.*

$$x_{bottom}, x_{top} = \{x_d : d \in bottom\}, \{x_d : d \in top\}$$

$$P(x_{top}, c|x, \theta) = \pi_c \prod_{d \in top} \theta_{cd}^{x_d} (1-\theta_{cd})^{1-x_d}$$

$$\frac{P(x|\theta,\pi)}{P(x_{top}|\theta,\pi)} = P(x_{bottom}|x_{top},\theta,\pi)$$

$$P(x_{bottom}|x_{top},\theta,\pi) \propto \frac{\sum_c P(x,c|\theta,\pi)}{\sum_c P(x_{top},c|\theta,\pi)}$$

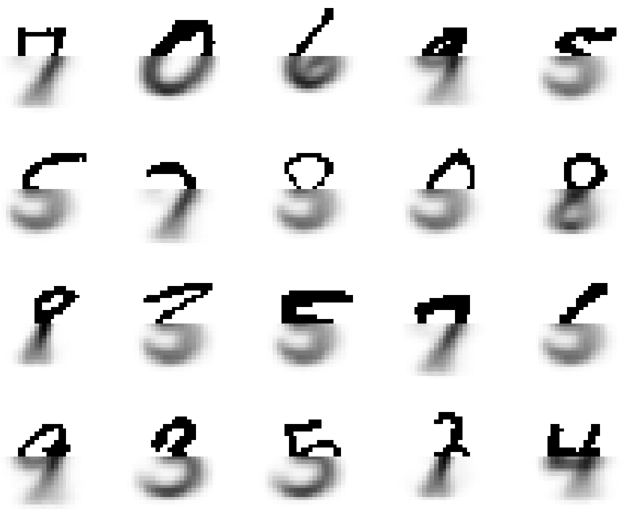
(e) **Derive $p(\mathbf{x}_{i\in bottom}|\mathbf{x}_{top},\theta,\pi)$, the marginal distribution of a single pixel in the bottom half of an image given the top half, conditioned on your fit parameters.**

$$\forall i \in bottom:$$

$$P(x_{i(bottom)}, x_{top}, c|\theta,\pi) = \pi_c \theta_{ci}^{x_i} (1-\theta_{ci})^{1-x_i} \prod_{d \in top} \theta_{cd}^{x_d} (1-\theta_{cd})^{1-x_d}$$

$$P(x_i|x_{top},\theta,\pi) = \frac{P(x_i, x_{top}|\theta,\pi)}{P(x_{top}|\theta,\pi)} = \frac{\sum_c P(x_i, x_{top}, c|\theta,\pi)}{\sum_c P(x_{top}, c|\theta,\pi)}$$

(f) **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.**



Problem 3 (Logistic Regression, 10 points)

Now, we'll fit a simple predictive model using gradient descent. Our model will be multiclass logistic regression:

$$p(c|\mathbf{x},\mathbf{w}) = \frac{\exp(\mathbf{w}_c^T\mathbf{x})}{\sum_{c'=0}^{9}\exp(\mathbf{w}_{c'}^T\mathbf{x})} \tag{4}$$

You can ignore biases for this question.

(a) **How many parameters does this model have?**

As we have 10 classes and there are 784 pixels per class, we have **7840** parameters in total.

(b) **Derive the gradient of the predictive log-likelihood w.r.t. w:** $\nabla_{\mathbf{w}} \log p(c|\mathbf{x},\mathbf{w})$

To simplify the operation consider the numerator **a** and denominator **z** respectively in $p(c|\mathbf{x},\mathbf{w})$. The computation of the partial derivative with respect to each single weight, w_{kd} is :
(note k signifies belonging to the right class and $\neg k$ in the wrong class)

$$\textit{Rigthful class}$$
$$\frac{\partial \log P(c|x,w)}{\partial w_{kd}} = \frac{z}{a}\frac{-x_d a^2 + x_d a z}{z^2} = x_d\left(1-\frac{a}{z}\right) = x_d - x_d P(k|x,w)$$

$$\textit{Wrongful class}$$
$$\frac{\partial \log P(c|x,w)}{\partial w_{\neg kd}} = \frac{z}{a}\frac{-x_d a * \exp(w_{\neg k}^T x)}{z^2} = -x_d\left(\frac{\exp(w_{\neg k}^T x)}{z}\right) = -x_d P(\neg k|x,w)$$

We construct the gradient by taking the partial derivative with respect to the individual classes and pixels using the above notions.

$$\nabla_w \log P(c|x,w) = .$$
$$\partial\, \textit{derivative with respect t an individual class}:$$
$$\begin{bmatrix} x_1 - x_1 P(0|x,w) & x_2 - x_2 P(0|x,w) & \cdots & x_{784} - x_{784} P(0|x,w) \end{bmatrix}$$
$$\begin{bmatrix} -x_1 P(c|x,w) & -x_2 P(c|x,w) & \cdots & -x_{784} P(c|x,w) \end{bmatrix}$$
$$\partial\, \textit{derivative with respect t pixel}:$$
$$\begin{bmatrix} x_1 - x_1 P(0|x,w) & -x_1 P(c|x,w) & \cdots & x_1 - x_1 P(9|x,w) \end{bmatrix}$$
$$\begin{bmatrix} x_{784} - x_{784} P(0|x,w) & -x_{784} P(c|x,w) & \cdots & x_{784} - x_{784} P(9|x,w) \end{bmatrix}$$

Problem 4 (Unsupervised Learning, 10 points)

Another advantage of generative models is that they can be trained in an unsupervised or semi-supervised manner. In this question, we'll fit the Naïve Bayes model without using labels. Since we don't observe labels, we now have a *latent variable model*. The probability of an image under this model is given by the marginal likelihood, integrating over c:

$$p(\mathbf{x}|\theta,\pi) = \sum_{c=1}^{k} p(\mathbf{x},c|\theta,\pi) = \sum_{c=1}^{k} p(c|\pi) \prod_{d=1}^{784} p(x_d|c,\theta_{cd}) = \sum_{c=1}^{k} Cat(c|\pi) \prod_{d=1}^{784} Ber(x_d|\theta_{cd}) \tag{5}$$

It turns out that this gives us a mixture model! This model is sometimes called a "mixture of Bernoullis", although it would be clearer to say "mixture of products of Bernoullis". Again, this is just the same Naïve Bayes model as before, but where we haven't observed the class labels c. In fact, we are free to choose K, the number of mixture components.

(a) Given K, how many parameters does this model have?

There are $K * (784 + pi_k)$ so **K*785**.

(b) Derive the gradient of the log marginal likelihood with respect to θ: $\nabla_\theta \log p(\mathbf{x}|\theta,\pi)$

The partial derivative(for theta cd) for the marginal likelihood:

$$\frac{\partial \log P(X|\theta,\pi)}{\partial \theta_{cd}} = \frac{\pi_c \theta_{cd}^{x_d}(1-\theta_{cd})^{1-x_{cd}}\left(\frac{x_d-\theta_{cd}}{\theta_{cd}(1-\theta_{cd})}\right)}{P(X|\theta,\pi)} * \left(\prod_{d\neq\forall j} \theta_{cj}^{x_j}(1-\theta_{cj})^{1-x_{cj}}\right)$$

(c) For a fixed $\pi_c = \frac{1}{K}$ and K = 30, fit θ on the training set using gradient based optimization. Note: you can't initialize at all zeros – you need to break symmetry somehow, which is done for you in starter code. Starter code reduces this problem to correctly coding the optimization objective. Plot the learned θ. How do these cluster means compare to the supervised model?