Assignment #4

Due: Never

In this assignment, we'll fit both generative and discriminative models to the MNIST dataset of handwritten numbers. Each datapoint in the MNIST dataset [`http://yann.lecun.com/exdb/mnist/`] is a 28x28 black-and-white image of a number in $\{0 \ldots 9\}$, and a label indicating which number.

MNIST is the 'fruit fly' of machine learning - a simple standard problem useful for comparing the properties of different algorithms. Some code for handling MNIST is on the course webpage.

You can use whichever programming language you like, and libraries for loading and plotting data. You'll need to write your own initialization, fitting, and prediction code. You can use "automatic differentiation" (not covered in lecture) in your code if you wish, but you should still answer the gradient questions.

For this assignment, we'll *binarize* the dataset, converting the grey pixel values to either black or white (0 or 1) with > 0.5 being the cutoff. When comparing models, we'll need a training and test set. Use the first 10000 samples for training, and another 10000 for testing. Hint: Also build a dataset of only 100 training samples to use when debugging, to make loading and training faster.

Question 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 each datapoint $\mathbf{x}$ and its class label c as follows:

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

where the vector $\boldsymbol{\pi}$ provides the prior probabilities for the classes and matrix $\boldsymbol{\theta}$ contains a set of parameters. For binary data, the likelihood $p(x_d|c, \theta_{cd})$ can have a Bernoulli distribution:

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

Which is just a way of expressing that $p(x_d = 1|c, \theta_{cd}) = \theta_{cd}$.

For $p(c|\boldsymbol{\pi})$, we can use a categorical distribution:

10 classes for the 10 digits

$$p(c|\boldsymbol{\pi}) = \text{Cat}(c|\boldsymbol{\pi}) = \prod_{i=0}^{9} \pi_i^{i=c} \tag{3}$$

categorical distribution -> generalized bernoulli

Which is a way of expressing that $p(c = t|\boldsymbol{\pi}) = \pi_t$. Note that we need $\sum_{i=0}^{9} \pi_i = 1$.

(a) Write down the *maximum a posteriori* estimate for the class-conditional pixel means $\boldsymbol{\theta}$, using a Beta(2, 2) prior on each θ_{cd}. Hint: it has a simple form, and you can ignore normalizing constants.

(b) Fit $\boldsymbol{\theta}$ to the training set. Plot $\boldsymbol{\theta}$ as 10 separate greyscale images, one for each class.

(c) Derive the predictive log-likelihood $\log p(c|\mathbf{x}, \boldsymbol{\theta}, \pi)$. Again you may ignore normalizing constants.

(d) Given parameters fit to the training set, and $\pi_c = \frac{1}{10}$ for all c, report the average predictive log-likelihood per datapoint, on both the training and test set. That is, select for each datum one of the 10 possible predictive log-likelihoods and then average over the dataset's 10k cases. Report the predictive accuracy on both the training and test set. The predictive accuracy is defined as the fraction of examples where the true class $t = \text{argmax}_c\, p(c|\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\pi})$. Remember to show your code for all questions in this assignment which involve coding.

Question 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: Any two pixels x_i and x_j where $i \neq j$ are independent given c.

(b) True or false: Any two pixels x_i and x_j where $i \neq j$ are independent when marginalizing over c.

(c) Using the parameters fit in question 1, produce random data samples from the model. That is, randomly sample and plot 10 binary images from the marginal distribution $p(\mathbf{x}|\boldsymbol{\pi}, \boldsymbol{\theta})$.

(d) Derive $p(\mathbf{x}_{\text{bottom}}|\mathbf{x}_{\text{top}}, \boldsymbol{\theta}, \boldsymbol{\pi})$, the joint distribution over the bottom half of an image given the top half, conditioned on your fit parameters. Hint: the relative class probabilities $p(c|\mathbf{x}_{\text{top}}, \boldsymbol{\theta}, \boldsymbol{\pi})$ will act as an information bottleneck.

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

(f) For 20 images from the training set, plot the top half of the image concatenated with the marginal distribution over each pixel in the bottom half.

Question 3 (Logistic Regression, 10 points)

Now, we'll fit a simple predictive model using gradient descent. Our model will be multiclass logistic regression as introduced in Lecture 5. Namely,

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

You can ignore biases for this question.

(a) How many parameters does this model have?

(b) Since class probabilities must sum to one, this imposes constraints on the predictions of our model. How many effective degrees of freedom does the model have? That is, what is the smallest number of parameters we could use to write an equally expressive model with a different parametrization?

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

(d) Code up the derivative, and a gradient-based optimizer of your choosing. Optimize $\mathbf{w}$ to maximize the log-likelihood of the training set, and plot the resulting parameters using one image per class. Since this objective is concave, you can initialize $\mathbf{w} = \mathbf{0}$. Using minibatches is optional. Hint: Use `scipy.logsumexp` or its equivalent to make your code more numerically stable.

(e) Given parameters fit to the training set, report both the average predictive log-likelihood per datapoint, and the predictive accuracy on both the training and test set. How does it compare to Naïve Bayes?

Question 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:

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

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?

(b) How many ways can we permute the parameters of the model ($\boldsymbol{\theta}$ and $\boldsymbol{\pi}$) and get the same marginal likelihood $p(\mathbf{x}|\boldsymbol{\theta})$? Hint: switching any two classes wont change the predictions made by the model about x_n.

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

(d) For a fixed $\pi_c = \frac{1}{K}\ \forall c$ and K = 30, fit $\boldsymbol{\theta}$ on the training set using gradient-based optimization. Note: you can't initialize with $\boldsymbol{\theta} = \mathbf{0}$ – you need to break symmetry somehow. Plot the learned $\boldsymbol{\theta}$. How do these cluster means compare to the supervised model?

(e) 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. Hint: You can reuse the formula for $p(\mathbf{x}_{i\in\text{bottom}}|\mathbf{x}_{i\in\text{top}},\boldsymbol{\theta},\boldsymbol{\pi})$ from before. How do these compare with the image completions from the supervised model?