Multivariate models; Generative classifiers

CPSC 440/550: Advanced Machine Learning

cs.ubc.ca/~dsuth/440/24w2

University of British Columbia, on unceded Musqueam land

2024-25 Winter Term 2 (Jan–Apr 2025)

Admin

admin

- Tutorials start this week
- Totally optional; you can go to any section
  - In the unlikely event that it's full, prioritize seats for those registered for that section
- Office hours calendar linked from Piazza – Thursday and Friday this week

- Basically everyone should be off the waitlist now; will get stragglers in too
  - This is just all a very manual process

- Quiz this week Wednesday–Saturday; schedule a slot now
  - A lot of you haven't!
- Don't leave assignment 1 to the last minute!
  - Due Friday 11:59pm (no longer 5pm)

Last time: MLE and MAP for Bernoulli model

- Bernoulli distribution: simple parameterized probability model for binary data
- If $X \sim \mathrm{Bern}(\theta)$, then for $x \in \{0, 1\}$ we have

$$\Pr(X = x \mid \theta) = \begin{cases} \theta & \text{if } x = 1 \\ 1 - \theta & \text{if } x = 0 \end{cases} = \theta^{\mathbb{1}(x=1)}(1-\theta)^{\mathbb{1}(x=0)} = \theta^x (1-\theta)^{1-x}$$

- Also write this as $p(x \mid \theta)$ or even $p(x)$, if context is clear
- Maximum likelihood estimate (MLE): $\arg\max_\theta p(\mathbf{X} \mid \theta)$, just $\hat{\theta} = n_1/n$
- Maximum a posteriori (MAP) estimate: adds a prior $p(\theta)$ to choose $\arg\max_\theta p(\theta \mid \mathbf{X}) = \arg\max_\theta p(\mathbf{X} \mid \theta) p(\theta)$
  - $\mathrm{Beta}(\alpha, \beta)$ prior acts like $\alpha - 1$ "fake" one observations, $\beta - 1$ "fake" zeros

Outline

1. Product of Bernoullis
2. Generative classifiers

Motivation: modeling traffic congestion

- We want to model traffic congestion in a big city
- Simple version: measure which intersections are busy on different days:

loc 1	loc 2	loc 3	loc 4	loc 5	loc 6	loc 7	loc 8	loc 9
0	1	0	1	1	1	0	0	1
0	0	1	1	0	0	0	0	0
0	1	0	1	1	1	0	0	1
0	1	0	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	0	0	1
0	1	0	1	1	1	1	1	0

- We'd like a model of this data, to identify problems or to route buses efficiently
  - "Location 4 is always busy," "location 1 is rarely busy"
  - "locations 7 and 8 are always the same," "location 2 is busy when location 7 is busy"
  - "There's a 25% chance you hit a busy spot if you take intersections 1 and 8"

Multivariate binary density estimation

- We can view this as multivariate binary density estimation:
  - Input: n iid samples of binary vectors $x^{(1)}, \ldots, x^{(n)}$ in $\{0,1\}^d$
  - Output: a model that can assign a probability to any binary vector $x \in \{0,1\}^d$

loc 1	loc 2	loc 3	loc 4	loc 5	loc 6	loc 7	loc 8	loc 9
0	1	0	1	1	1	0	0	1
0	0	1	1	0	0	0	0	0
0	1	0	1	1	1	0	0	1
0	1	0	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	0	0	1
0	1	0	1	1	1	1	1	0

$\xrightarrow{\text{estimator}}$

$$p((0,\ldots,0)) = 0.0001$$

$\vdots$ (2^9 total values) $\vdots$

$$p((1,\ldots,1)) = 0.002$$

- Another example: "are there >10% covid cases in area j?"
- Notation (memorize):
  - We have n examples, each with d number of features
  - $x^{(4)}$ is a vector of length d, with elements $x_1^{(4)}$ to $x_d^{(4)}$
  - X_3 is the third dimension of a random vector X; x_3 is a value X_3 might take

Product of Bernoullis model

- There are many possible models for binary density estimation
  - Each one makes different assumptions; we'll see lots of options
- We'll start with a very simple "product of Bernoullis" model
  - Here we assume that all the dimensions are independent
    - If $d = 4$, this means $p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2)p(x_3)p(x_4)$
- We treat our d dimensional problem as d separate univariate problems

$$\mathbf{X} = \begin{array}{ccc} \text{loc 1} & \text{loc 2} & \text{loc 3} \\ \hline 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{array} \xrightarrow{\text{reframe}} \mathbf{X}_1 = \begin{array}{c} \text{loc 1} \\ \hline 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{array} \quad \mathbf{X}_2 = \begin{array}{c} \text{loc 2} \\ \hline 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{array} \quad \mathbf{X}_3 = \begin{array}{c} \text{loc 3} \\ \hline 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{array}$$

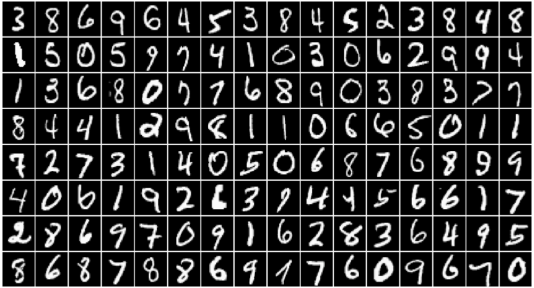
Product of Bernoullis: inference and learning

- Advantage of doing this: it makes inference and learning really easy
- For most tasks: just do it on each variable, then combine results
- Joint probability: $\Pr(X_1{=}1, X_2{=}0, \ldots, X_d{=}1) = \Pr(X_1{=}1)\Pr(X_2{=}0)\cdots\Pr(X_d{=}1) = \theta_1(1-\theta_2)\cdots\theta_d$
- Marginal probability: $\Pr(X_2{=}1) = \theta_2$, $\Pr(X_2{=}1, X_3{=}1) = \Pr(X_2{=}1)\Pr(X_3{=}1) = \theta_2\theta_3$
- Conditional probabilities: $p(x_2 \mid x_3) = p(x_2)$
- Mode: set x_1 from $\arg\max_{x_1} p(x_1)$, . . . , x_d from $\arg\max_{x_d} p(x_d)$
- Sampling: sample x_1 from $p(x_1)$, . . . , x_d from $p(x_d)$

- MLE: $\hat{\theta}_1 = \frac{n_{11}}{n} = \frac{\text{number of times } X_1 \text{ is } 1}{n}$, . . . , $\hat{\theta}_d = \frac{n_{d1}}{n}$; MAP is similar
  - `np.mean(X, axis=0)`; takes $\mathcal{O}(nd)$ time
    - Or $\mathcal{O}(\texttt{nnz}(X))$ if X is a sparse matrix with $\texttt{nnz}(X) \leq nd$ nonzero entries

Running example: MNIST digits

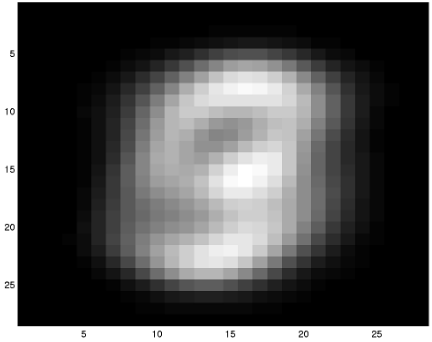
- We'll often use a basic dataset, MNIST digits, as an example
  - $n = 60,000$ images; each is a 28×28 grayscale image of a handwritten number
  - For binary density estimation: $d = 784$ for each pixel, rounded to $\{0, 1\}$



- In CPSC 340, we wanted a function that takes in an image and says "this is a 7"
- In density estimation, we want a probability distribution over images
  - What's the probability that some 28×28 grayscale image is a handwritten digit?
  - Unsupervised density estimation (ignoring the class label) for now
  - Sampling from the density should produce a novel image of a digit

Product of Bernoullis on MNIST

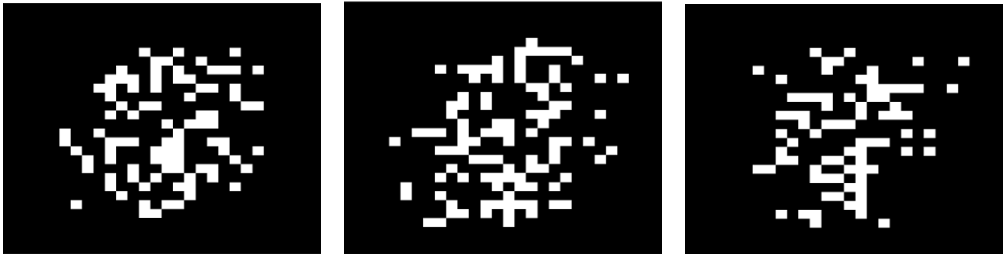
- If we fit a product of Bernoullis to MNIST:
  - Have 784 parameters: each pixel location is $\mathrm{Bern}(\theta_j)$
  - The MLE $\hat{\theta}_j$ is just the portion of the time pixel j "has ink"
- Viewing the $\hat{\theta}_j$ shaped into an image:



- More likely to have writing near the centre of the image

Product of Bernoullis on MNIST

- Is this a good fit to MNIST?
- One way to check: look at samples from the model

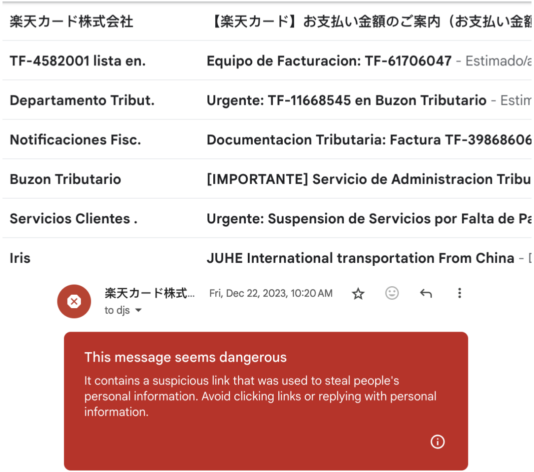


- This is a terrible model – the sample don't look like the data at all
- In the data, the pixels are far from independent
  - For example, adjacent pixels are highly correlated with each other
- Even though the assumption is usually wrong,
  a product of Bernoullis is often "good enough to be useful"
  - Especially as a component of a model, instead of the whole thing
- We'll see several ways to relax the independence assumption later in the course

Outline

Motivation: spam filtering

- Spam used to be a huge problem, until ML-based spam detectors



- Can frame as supervised learning
  - Learn a function from e-mails to “is this spam”

Data collection and feature extraction

review

- Collect a lot of emails
- Get users to label them as spam ($y^{(i)} = 1$) or not ($y^{(i)} = 0$)
- Extract features of each email, e.g. bag of words: $x_j^{(i)} = \mathbb{1}(\text{word } j \text{ in email } i)$

`winner`	`CPSC`	`440`	`vicodin`	...	spam?
1	0	0	0	...	1
0	1	1	0	...	0
0	0	0	1	...	1
1	1	1	0	...	0

- $y^{(i)}$ is label of ith example; collected in vector $\mathbf{y}$ (length n)
- $x_j^{(i)}$ is jth feature of ith example
- $x^{(i)}$ is the vector (length d) of features for ith example
- $\mathbf{X}$ collects all the inputs, shape $n \times d$ – in practice, use a sparse format!
- X_j is random variable for the jth feature of random sample; X is random vector

Generative classifiers

- Early '00s: best spam filtering methods used generative classifiers
- Treat supervised learning as density estimation
- Learning: fit a model for $p(x_1, \ldots, x_d, y)$
    - "Data-generating process" for the features and labels together
- Inference: compute conditionals $p(y \mid x_1, \ldots, x_d)$
    - Is $p(y = 1 \mid x_1, \ldots, x_d) > p(y = 0 \mid x_1, \ldots, x_d)$?

- Should we plug in our new fancy product of Bernoullis as our density estimator?
- Probably not – it assumes Y is independent of $X_1, \ldots, X_d$!
- So predictions wouldn't depend on the features at all!

Naïve Bayes

- Product of Bernoullis assumes $X_1, \ldots, X_d, Y$ are all mutually independent
- Naïve Bayes assumes x_j are mutually independent given y

- $X_1, \ldots, X_d$ are (mutually) independent if

$$p(x_1, \ldots, x_d) = p(x_1) \cdots p(x_d) \quad \text{for all possible values } x_1, \ldots, x_d$$

- $X_1, \ldots, X_d$ are (mutually) conditionally independent given y if

$$p(x_1, \ldots, x_d \mid y) = p(x_1 \mid y) \cdots p(x_d \mid y) \quad \text{for all possible values } x_1, \ldots, x_d, y$$

- Features independent per class: use a different product of Bernoullis for each class
- To fit, need conditional univariate density estimates

$$p(x_1, \ldots, x_d, y) = p(x_1, \ldots, x_d \mid y)p(y) = p(x_1 \mid y) \cdots p(x_d \mid y)p(y)$$

Naïve Bayes inference

- Given model, can compute $p(x_1, \ldots, x_d, y) = p(x_1 \mid y) \cdots p(x_d \mid y) p(y)$
- To classify: have $p(y \mid x) \propto p(x, y)$ – probabilistic predictions
  - We maximize probability of correct answer if we take $\arg\max_y p(y \mid x)$
- But probabilities make it easy to do more variations!
- Probably cost of missing a spam email $<$ cost of flagging a non-spam email

Prediction \Truth	$y = 0$: Good email	$y = 1$: Spam
$\hat{y} = 0$: Good email	0	1
$\hat{y} = 1$: Spam	50	0

- Can minimize expected cost: letting $\rho(x) = p(y = 1 \mid x)$ be the prediction,

$$\mathbb{E}[C(\hat{y}, y)] = \rho(x)\, C(\hat{y}, 1) + (1 - \rho(x))\, C(\hat{y}, 0)$$

$$= \begin{cases} (1 - \rho(x)) \cdot 0 + \rho(x) \cdot 1 & \text{if } \hat{y} = 0 \\ (1 - \rho(x)) \cdot 50 + \rho(x) \cdot 0 & \text{if } \hat{y} = 1 \end{cases} = \begin{cases} \rho(x) & \text{if } \hat{y} = 0 \\ 50(1 - \rho(x)) & \text{if } \hat{y} = 1 \end{cases}$$

so we predict $\hat{y} = 1$ only if $50(1 - \rho(x)) \leq \rho(x)$, i.e. $\rho(x) \geq \frac{50}{51} \approx 98\%$

Naïve Bayes inference

- Can also do other inference tasks:

- What's $p(x_1, \ldots, x_d)$?
- What are the "most spammy" features? $\arg\max_{x_1,\ldots,x_d} p(x_1, \ldots, x_d \mid y = 1)$
- How can I minimally change my spam email to make it look not like spam?
- Generate data with ancestral sampling:
  - Sample $\tilde{y}$ from $p(y)$, then $\tilde{x}$ from $p(x \mid \tilde{y})$

Training naïve Bayes: conditional binary density estimation

- Recall that under naïve Bayes assumption,

$$p(x_1, \ldots, x_d, y) = p(x_1, \ldots, x_d \mid y)p(y) = p(x_1 \mid y) \cdots p(x_d \mid y)p(y)$$

- For binary X_j and Y: $p(y)$ is just binary density estimation
- Can parameterize $p(x_j \mid y) = \Pr(X_j = x_j \mid Y = y)$ as conditionally Bernoulli:

$$\Pr(X_j = 1 \mid Y = 1) = \theta_{j|1} \qquad \Pr(X_j = 1 \mid Y = 0) = \theta_{j|0}$$

- Two parameters per feature: $\theta_{j|y}$ is probability of X_j being 1 given $Y = y$
- $X_j \mid Y = 0$ is $\mathrm{Bern}(\theta_{j|0})$, and $X_j \mid Y = 1$ is $\mathrm{Bern}(\theta_{j|1})$
- MLE is given by "counting conditionally":

$$\hat{\theta}_{j|1} = \frac{\sum_{i=1}^{n} \mathbb{1}(x_j^{(i)} = 1)\, \mathbb{1}(y^{(i)} = 1)}{n_1} = \frac{n_{x_j=1,y=1}}{n_{y=1}} \qquad \hat{\theta}_{j|0} = \frac{n_{x_j=1,y=0}}{n_{y=0}}$$

  - Should be intuitive, but worth writing out for yourself to check the steps make sense!

Generative classifiers

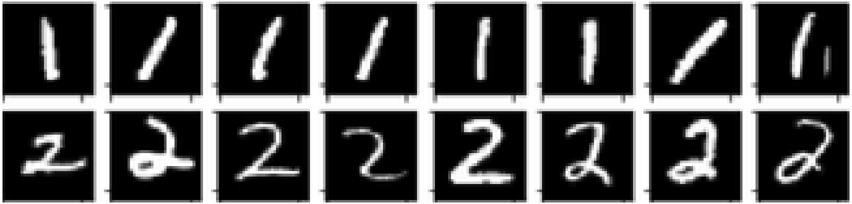
- Training phase: density estimation: fit a model for $p(x, y)$
- Usually: first fit a model for $p(y)$
  - For binary y, just use a Bernoulli and do MLE/MAP $\mathcal{O}(n)$ time
- Next, for each class c, fit $p(x \mid y = c)$ using examples from class c
  - For naïve Bayes, fit $p(x_1 \mid y = c), \ldots, p(x_d \mid y = c)$ separately
  - For binary data, fits a product of Bernoullis for class c $\mathcal{O}(n_{y=c}\, d)$ time

- Total: $\mathcal{O}(n + n_{y=1}d + \cdots + n_{y=k}d) = \mathcal{O}(n + nd) = \mathcal{O}(nd)$ time
  - Can reduce to $\mathcal{O}$(number of nonzero entries) with sparse format

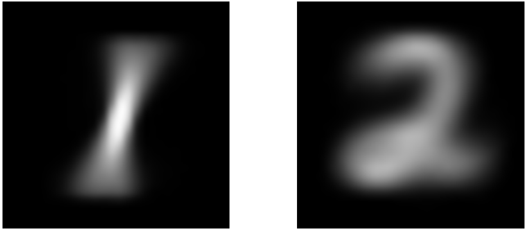
- Testing phase: use $p(y \mid x) \propto p(x, y)$ to get probability of each class for x
- Usually: predict $\arg\max_y p(y \mid x) = \arg\max_y p(x, y)$
  - "What's the most likely y, after seeing x?" (Like MAP!)

Naïve Bayes on MNIST

- Let's make a binary supervised learning problem: distinguish 1 from 2



- There are 6,742 1s, and 5,958 2s
  - With MLE, get $p(y = 1) = 6\,742/(6\,742 + 5\,958) \approx 0.53$
- MLE parameters for Naïve Bayes, $p(x_j \mid y)$ for each class (arranged as an image):



Naïve Bayes on MNIST

- Sample class $\tilde{y}$ from $p(y)$, then features from $p(x \mid \tilde{y})$:

- Clearly different from the dataset, but at least there's some structure
- We don't need a perfect model for naïve Bayes to classify well
  - Might be enough to see 2 is more likely than 1, even if it's a bad model of each class
  - Naïve Bayes is a terrible estimator of email distribution, but "good enough" classifier

Summary

- Product of Bernoullis:
  - Extremely simple way to handle multivariate data
  - Assumes all variables are independent
  - Very strong assumption gives really easy inference/learning but bad models
- Generative classifiers: model $p(x, y)$, then use $p(y \mid x)$ to classify
- Naïve Bayes: assume that dimensions of x are independent given y

- Next time: discriminating (but in a good way)