We introduced the beta distribution:

\[ p(\theta | \alpha, \beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)} \propto \theta^{\alpha-1}(1-\theta)^{\beta-1} \]

- Which is a probability over a parameter \( \theta \) in the range [0,1].

We reviewed the “\( \propto \)" notation for probabilities:
- If \( p(\theta) \propto g(\theta) \) for discrete \( \theta \), then \( p(\theta) = g(\theta)/\sum_{\theta'} g(\theta') \).
- If \( p(\theta) \propto g(\theta) \) for continuous \( \theta \), then \( p(\theta) = g(\theta)/\int g(\theta')d\theta' \).

For Bernoulli likelihood and beta prior, we showed posterior is:

\[ p(\theta | X, \alpha, \beta) = \frac{\hat{\theta}^{\alpha-1}(1-\hat{\theta})^{\beta-1}}{\hat{B}(\hat{\alpha}, \hat{\beta})} \propto \hat{\theta}^{\alpha-1}(1-\hat{\theta})^{\beta-1} \]

- Where \( \hat{\alpha} = \alpha + n_1 \) and \( \hat{\beta} = \beta + n_0 \).
- It looks like the prior, with \( \alpha \) and \( \beta \) “updated” by the counts of 1s and 0s from data.
MAP Estimation for Bernoulli-Beta Model

- The posterior with a Bernoulli likelihood and beta prior is a beta:

\[
\rho(\theta | X, \alpha, \beta) = \frac{\theta^{\tilde{\alpha} - 1} (1-\theta)^{\tilde{\beta} - 1}}{\text{beta function}(\tilde{\alpha}, \tilde{\beta})}
\]

- Where \(\tilde{\alpha} = n_1 + \alpha\) and \(\tilde{\beta} = n_0 + \beta\).

- If \(\tilde{\alpha} > 1\) and \(\tilde{\beta} > 1\), taking log and setting derivative to 0 gives MAP of:

\[
\hat{\theta} = \frac{n_1 + \alpha - 1}{n_1 + \alpha - 1 + n_0 + \beta - 1}
\]

- If \(\alpha = 1\) and \(\beta = 1\), we get the MLE.
- If \(\alpha = 2\) and \(\beta = 2\), we get Laplace smoothing (which often overfits less).
- If \(\alpha = \beta > 2\), we get a stronger bias towards \(\hat{\theta} = 0.5\) than Laplace smoothing.
- If \(\alpha = \beta < 1\), we get a bias towards away from \(\hat{\theta} = 0.5\) (towards 0 or 1).
- You can also bias towards either 0 or 1; if \(\alpha\) is large compared to \(\beta\) it biases towards \(\hat{\theta} = 1\).
- Notice that MAP converges to MLE \(n \to \infty\), so the data eventually "takes over" estimate.

- But we use a prior so our model does sensible things when we do not have enough data.
Review: Hyper-Parameter and [Cross]-Validation

• We call the “parameters of the prior”, $\alpha$ and $\beta$, the hyper-parameters.
  – We usually say that hyper-parameters are “parameters affecting the complexity of the model”.
  – We usually also include “parameters of the learning algorithm” as hyper-parameters.

• How can we choose hyper-parameters values?
  – Using the training likelihood does not work: it would make $\alpha$ and $\beta$ arbitrarily small (ignoring prior).

• Usual CPSC 340 approach: use a validation set (or cross-validation).
  – Split your data ‘X’ into a “training” set and a “validation” set.
  – For different hyper-parameters of $\alpha$ and $\beta$:
    • Use the “training” examples to compute the MAP estimate.
    • Use MAP estimate to compute the likelihood of the “validation” examples.
  – Choose the hyper-parameters with the highest validation likelihood.
    • But our final goal is to not optimize performance on the validation set.
    • This is a surrogate for the test error (error on completely-new data), which you cannot measure.

• Take CPSC 340 to learn about many of the things that can go wrong.
  – For example, if you are not careful you can overfit to the validation set.
    • I see this all the time, even in UBC student’s PhD theses!
Next Topic: Product of Bernoullis
Motivation: Modeling Traffic Congestion

• We want to model car “traffic congestion” in a big city.
• So we measure which intersections are busy on different days:

<table>
<thead>
<tr>
<th>Inter 1</th>
<th>Inter 2</th>
<th>Inter 3</th>
<th>Inter 4</th>
<th>Inter 5</th>
<th>Inter 6</th>
<th>Inter 7</th>
<th>Inter 8</th>
<th>Inter 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

• We want to build a model of this data, to identify patterns/problems.
  – “Inter 4 is always busy”, “Inter 1 is rarely busy”.
  – “Inters 7+8 are always the same”, “Inter 2 is busy when Inter 7 is busy”.
  – “There is a 25% chance you will hit a busy intersection if you take Inter 1 and 8”.
Problem: Multivariate Binary Density Estimation

- We can view this as **multivariate** binary density estimation:
  - Input: ‘n’ IID samples of **binary vectors** \(x^1, x^2, x^3, \ldots, x^n\) from population.
  - Output: model that gives probability for any assignment of values \(x_1, x_2, \ldots, x_d\).

- Covid example: each feature could be “are covid cases >10% in area ‘j’?”

- Notation (please memorize):
  - We use ‘\(n\)’ for the number of examples, ‘\(d\)’ for the number of features.
  - Notice that \(x^3\) is a vector with ‘\(d\)’ values, and \(x_3\) is one binary value.

\[
\begin{array}{cccccccccc}
\text{Inter 1} & \text{Inter 2} & \text{Inter 3} & \text{Inter 4} & \text{Inter 5} & \text{Inter 6} & \text{Inter 7} & \text{Inter 8} & \text{Inter 9} \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]

\[
p(x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 1, x_6 = 1, x_7 = 0, x_8 = 0, x_9 = 1) = 0.11
\]

(Estimates probability for all \(2^9\) values)
Product of Bernoullis Model

• There are many different models for binary density estimation.
  – Each one makes different assumptions, we will see lots of options!
• We will first consider the simple “product of Bernoullis” model.
  – In this model we assume that the variables are “mutually independent”.
  • So if we have four variables we assume

\[ p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2)p(x_3)p(x_4). \]
  – As a picture, we treat multivariate problem as ‘d’ univariate problems:

<table>
<thead>
<tr>
<th>Inter 1</th>
<th>Inter 2</th>
<th>Inter 3</th>
<th>Inter 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \Theta_1 \\ \Theta_2 \\ \Theta_3 \\ \Theta_4 \end{bmatrix} \]
Product of Bernoullis Inference and Learning

• Key advantage of “product of Bernoullis” model: easy inference and learning.
  – For most inference tasks: do inference on each variable, then combine the results.
  – Compute joint probability.
    • \( p(x_1, x_2, \ldots, x_d) = p(x_1)p(x_2) \cdots p(x_d) = \theta_1 \theta_2 \cdots \theta_d \).
  – Compute marginal probabilities.
    • \( p(x_2) = \theta_2 \).
    • \( p(x_2, x_3) = p(x_2)p(x_3) = \theta_2 \theta_3 \).
  – Compute conditional probabilities.
    • \( p(x_2 \mid x_3) = p(x_2) \).
    • \( p(x_2, x_3 \mid x_4) = p(x_2, x_3) = \theta_2 \theta_3 \).
  – Decoding of \( p(x_1, x_2, \ldots, x_d) \): set \( x_1 \) to argmax value of \( p(x_1) \), set \( x_2 \) to argmax of \( p(x_2) \),..., set \( x_d \) to argmax value of \( p(x_d) \).
  – Sampling:
    • Sample \( x_1 \) from \( p(x_1) \), sample \( x_2 \) from \( p(x_2) \),..., sample \( x_d \) from \( p(x_d) \).

• MLE (MAP is similar):
  \[ \hat{\theta}_1 = \frac{n_{1i}}{n} \left\rightharpoonup \text{number of times variable } i \text{ is } 1 \right\]
  \[ \hat{\theta}_2 = \frac{n_{2i}}{n} \cdots \hat{\theta}_d = \frac{n_{di}}{n} \]
Product of Bernoullis Inference and Learning

• MLE in a product of Bernoullis:

\[
\theta = \text{zeros}(d) \\
\text{for } i \text{ in } 1:n \\
\quad \text{for } j \text{ in } 1:d \\
\quad \quad \text{if } X[i,j] = 1 \\
\quad \quad \quad \theta[j] += 1 \\
\theta ./= n
\]

or

\[
\theta = \frac{\text{sum}(X, \text{dims}=1)}{n}
\]

– Sum up columns of 'X'

– Count the number of times each \(x_j=1\)

– Divide by 'n'

• Cost is \(O(nd)\): do an \(O(1)\) operations \(n*d\) times, then \(O(n)\) division.
  – If ‘X’ is stored as a “sparse” matrix, can be implemented to only cost \(O(z)\).
    • Where ‘z’ is the number of non-zero values (\(z \leq nd\)).

• Sampling code:

\[
\begin{align*}
\mathbf{x} &= \text{zeros}(d) \\
\text{for } j \text{ in } 1:d \\
\quad x[j] &= \text{SampleBinary}(\theta[j])
\end{align*}
\]

• Cost is \(O(d)\) to generate a sample.
Running Example: MNIST Digits

• To illustrate density estimation, we will often use the MNIST digits:
  – 60,000 images, each a 28x28 pixel image of a number.
  – Representing as binary density estimation:
    • Each image is one training example $x_i$.
    • With each feature being one of the 784 pixels.
    • Threshold each pixel to make it binary.

• CPSC 340 wanted to “recognize that this is a 4”.

• In density estimation we want probability distribution over images.
  – Given one of the $2^{784}$ possible images, what is the probability it is a digit?
    • This is unsupervised, we are ignoring the class labels.
  – Sampling from the density should generate new images of digits.

Product of Bernoullis on MNIST Digits

• Consider fitting the **product of Bernoullis model to MNIST digits**:
  – For each of the 784 pixels ‘j’, we will have a parameter $\theta_j$.
    • A “position-specific Bernoulli” distribution.
  – To compute MLE for $\theta_j$, compute fraction of times pixel ‘j’ was set to 1.
    • Visualizing those MLE values as an image:

  ![Image](image.png)

  • Shows pixels near the center are more likely to be 1 than pixels near the boundary.
Product of Bernoullis on MNIST Digits

• Is product of Bernoullis a good model for the MNIST digits?
  – Samples generated from the model (independent sample from position-specific Bernoulli for each pixel):

  – This is a terrible model: these samples do not look like the data.
  – Why is this a terrible model?
    • In the dataset, the pixels are not independent.
    • For example, pixels that are “next” to each other in the image are highly correlated.
  – Even it is a bad model, product of Bernoullis is often “good enough to be useful”.
    • Usually when combined with other ideas, that we will see shortly.
    • In practice, I think it is actually the most-used method for binary density estimation even though it is one of the worst.
  – Later in the course will cover several ways to relax the independence assumption.
Next Topic: Generative Classifiers
Motivation: E-mail Spam Filtering

• Want a build a system that detects spam e-mails.
  – Context: spam used to be a big problem.

• We can write this as a supervised learning problem:
  – Want to learn to map from “input” (e-mail) to “output” (spam or not).
Review: Data Collection and Feature Extraction

• Collect a large number of e-mails, gets users to label them.

<table>
<thead>
<tr>
<th></th>
<th>$</th>
<th>Hi</th>
<th>CPSC</th>
<th>340</th>
<th>Vicodin</th>
<th>Offer</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>…</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>…</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

• We can use \( y_i = 1 \) if e-mail \( i \) is spam, \( y_i = 0 \) if e-mail is not spam.

• Extract features of each e-mail (like “bag of words”).
  – \( x_{ij} = 1 \) if word/phrase ‘j’ is in e-mail ‘i’, \( x_{ij} = 0 \) if it is not.

• See CPSC 330 (or 340) for different ways to extract features from text data.
Our notation for supervised learning:

- **X** is matrix of all features, **y** is vector of all labels.
  - We use \( y^i \) for the label of example ‘\( i \)’ (element ‘\( i \)’ of ‘\( y \)’).
  - We use \( x^i_j \) for feature ‘\( j \)’ of example ‘\( i \)’.
  - We use \( x^i \) as the list of features of example ‘\( i \)’ (row ‘\( i \)’ of ‘\( X \)’).
    - So in the above \( x^3 = [0 \ 1 \ 1 \ 0 \ 0 \ ...] \).
    - In practice, store \( x^i \) in some “sparse” format (like a list of non-zeroes, smaller memory).
Generative Classifiers

• In early 2000s, best spam filtering methods used generative classifiers.
  – Generative classifiers treat supervised learning as density estimation.

• How can we do supervised learning with density estimation?
  – Learning: use a density estimator to estimate \( p(x_1, x_2, \ldots, x_d, y) \).
    • Generative classifiers model “how the features and label were generated”.
  – Inference: compute conditionals \( p(y | x_1, x_2, \ldots, x_d) \) to make predictions.
    • For example, is \( p(y = 1 | x_1, x_2, \ldots, x_d) > p(y = 0 | x_1, x_2, \ldots, x_d) \)?

• Can we use a product of Bernoullis as the density estimator?
  – You could, but it would do terrible!
  – If ‘y’ is independent of the features, predictions would ignore features.
  – A simple model that does assume ‘y’ is independent of features is naïve Bayes.
Summary

• **Beta distribution:**
  – Prior for Bernoulli parameter that yields a closed-form MAP estimate.
    • Laplace smoothing as a special case.

• **Hyper-parameters:**
  – Parameters of the prior, or other parameters affecting complexity.
  – Later we will also include “parameters of the optimization algorithm”.

• **Product of Bernoullis:**
  – Method for multivariate binary density estimation.
  – Assumes all variables are independent.
  – Inference and learning are easy, but cannot accurate model many densities.

• **Generative classifiers:**
  – Classifiers that model $p(x,y)$ and predict by doing inference.

• Next time: a bit of 340 review.
Existence of MAP Estimate under Beta Prior

• The MAP estimate for Bernoulli likelihood and beta prior:

\[
\hat{\theta} = \frac{n_1 + \alpha - 1}{(n_1 + \alpha - 1) + (n_0 + \beta - 1)}
\]

– This assumes that \( n_1 + \alpha > 1 \) and \( n_0 + \beta > 1 \).

• Other cases:
  – \( n_1 + \alpha > 1 \) and \( n_0 + \beta \leq 1 \): \( \hat{\theta} = 1 \).
  – \( n_1 + \alpha \leq 1 \) and \( n_0 + \beta > 1 \): \( \hat{\theta} = 0 \).
  – \( n_1 + \alpha < 1 \) and \( n_0 + \beta < 1 \): \( \hat{\theta} \) can be 0 or 1.
  – \( n_1 + \alpha = 1 \) and \( n_0 = 1 \): \( \hat{\theta} \) can be anything between 0 and 1.