Introduction to Machine Learning (IML)

LECTURE #11: GENERATIVE MODELS

236756 – 2023-2024 WINTER – TECHNION

LECTURER: YONATAN BELINKOV

Today

- **part III**: *more supervised learning*
  1. ~~Regression~~
  2. ~~Bagging and Boosting~~
  3. Generative models
  4. Deep learning

Types of Data Distributions

Joint:
$(x, y) \sim D_{XY}$
(sample input-label pair)

Marginal:
$x \sim D_X$
(sample input)
data uncertainty

Conditional:
$y \sim D_{Y|X=x}$
(sample label for given input)
label uncertaint(ies)

- Which distributions should we care about?

Recap: Discriminative Approach

- Recall, our data come from a joint distribution $(x, y) \overset{iid}{\sim} D_{XY}$
- But we focused on modeling $P(y|x) \coloneqq P(Y = y|X = x)$ directly
- This allows us to predict
  - E.g., in binary classification, given x:

$$h(x) = \begin{cases} 1 & P(y = 1|x) > \frac{1}{2} \\ 0 & otherwise \end{cases}$$

- This is called a discriminative approach
- **Remember**: we parameterize this with some θ as $P(y|x;\theta)$
  - E.g., for linear classifiers $\theta = w$ and we have $P(y|x;\theta) = w^T x$

Generative Approach

- Recall Bayes' rule:

$$P(y|x) = \frac{P(x,y)}{P(x)} = \frac{P(x|y)P(y)}{\sum_{y'} P(x|y')P(y')}$$

- This gives a classifier:

$$h(x) = \text{argmax}_y \frac{P(x|y)P(y)}{\sum_{y'} P(x|y')P(y')} = \text{argmax}_y P(x|y)P(y)$$

Class-Conditional Distribution

Class Prior

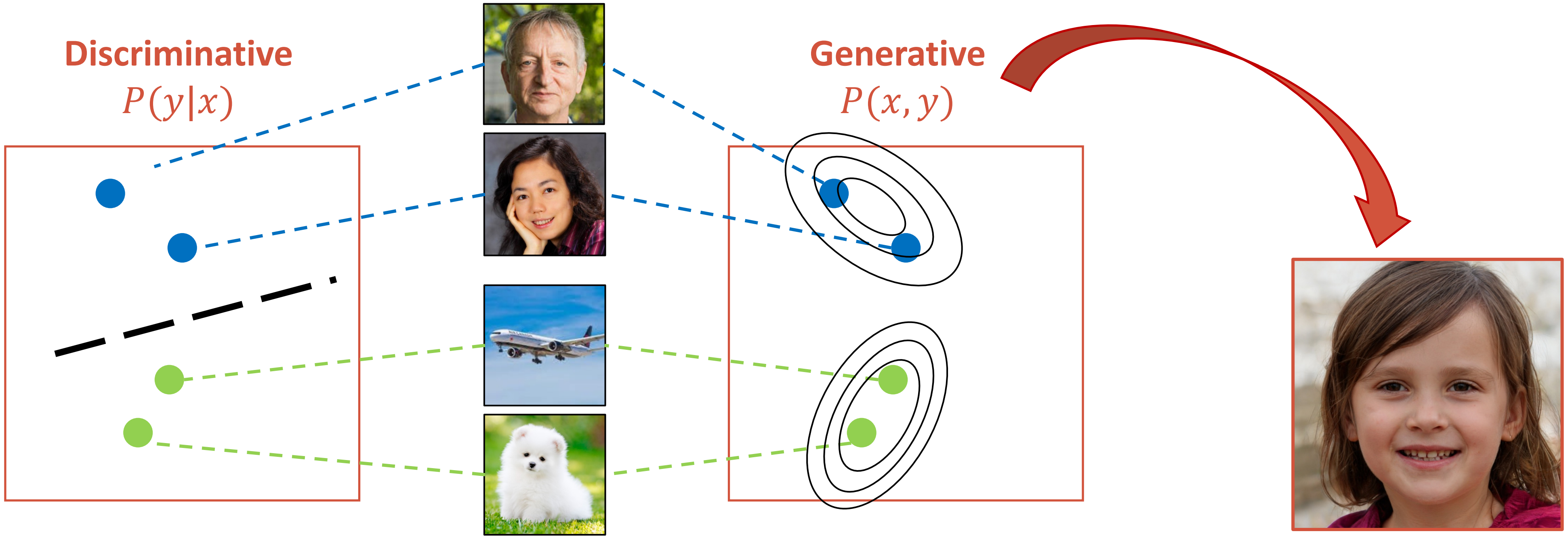
- Generative approach: Estimate $P(x|y)$ and $P(y)$
  - Implies estimating $P(y|x)$ because $P(y|x) \propto p(x|y)p(y)$
- Again, these have some parameters θ:

$$P(y|x;\theta) \propto P(x|y;\theta)P(y;\theta) = P(x,y;\theta)$$

Generative vs. Discriminative Models

- Generative models solve a harder task
- But sometimes it's easy to learn their parameters
- And modeling the data can be useful

"When solving a given problem, try to avoid a more general problem as an intermediate step"



Sources: Google's ML crash course, ThisPersonDoesNotExist

Generative stories

- Generative models tell a generative story about the data generation
- Think of our Papaya
  - First, we sample a label y from Y according to $P(Y = y; \theta)$
  - Then, we sample x from $X|Y = y$ according to $P(X = x|Y = y; \theta)$

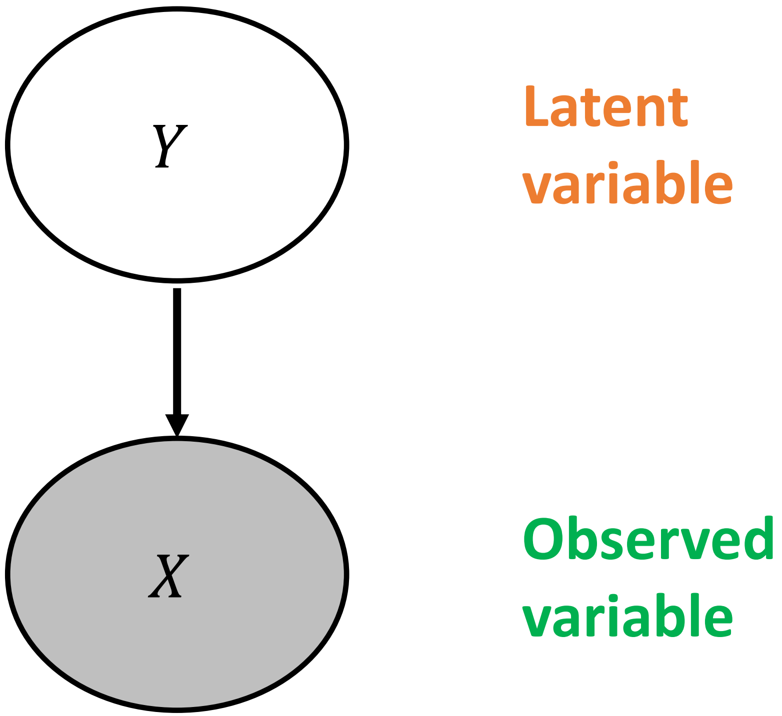
prior → **yummy/not** → 

- The core of modeling is in deciding on parametric distributions

Graphical models

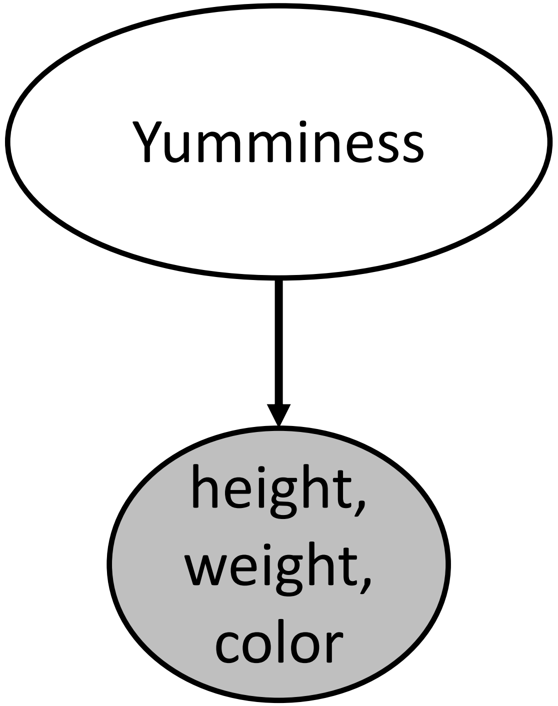
- It is often convenient to represent the data generation process in a **graphical model**
- Arrows represent conditional dependencies $P(x|y;\theta)$
- Can read the data probability from the graph

$$P(x, y; \theta) \\ = P(y;\theta)P(x|y;\theta)$$



Graphical models

- It is often convenient to represent the data generation process in a **graphical model**



Training generative models

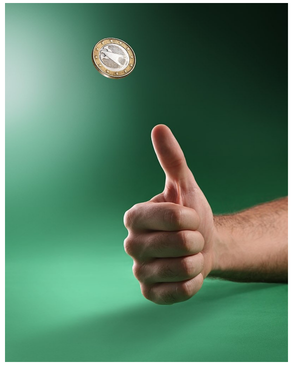
- How to estimate $P(x|y;\theta)$ and $P(y;\theta)$?
- Enter Maximum Likelihood Estimation (MLE)
  - For **training set $\boldsymbol{S}$** $= \{(x_i, y_i)\}_{i=1}^m$, seek parameters $\hat{\theta}$ that maximize the log-likelihood: $\log L(\theta; S) = \log P(S;\theta)$

$$\log L(\theta;S) = \log P(S;\theta) = \log \mathrm{P}(\{(x_i, y_i)\}_{i=1}^m;\theta) = \log \prod_i P(x_i, y_i;\theta)$$

$$= \log \prod_i P(x_i|y_i;\theta) p(y_i;\theta) = \sum_i \log P(x_i|y_i;\theta) + \sum_i \log P(y_i;\theta)$$

- Result: can optimize parameters for each term separately
  - They have different θ's
- Let's see some examples

Example: Biased coin

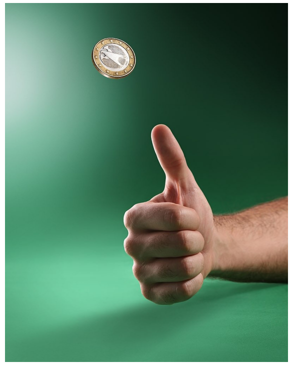


- For now, forget about y, and consider a set of coin flips $S = \{x_i\}_{i=1}^m$, where the probability of heads is $\theta^* = P(\mathrm{H})$ (H=HEADS, T=TAILS)
- What is $\hat{\theta} = \mathrm{argmax}_\theta \log L(\theta; S)$?

$$\log L(\theta; S) = \log P(S; \theta) = \log \mathrm{P}(\{x_i\}_{i=1}^m; \theta) = \log \prod P(x_i; \theta) = \sum_i \log P(x_i; \theta)$$

https://en.wikipedia.org/wiki/Coin_flipping

Example: Biased coin



- For now, forget about y, and consider a set of coin flips $S = \{x_i\}_{i=1}^{m}$, where the probability of heads is $\theta^* = P(\mathrm{H})$ (H=HEADS, T=TAILS)
- What is $\hat{\theta} = \mathrm{argmax}_\theta \log L(\theta; S)$?

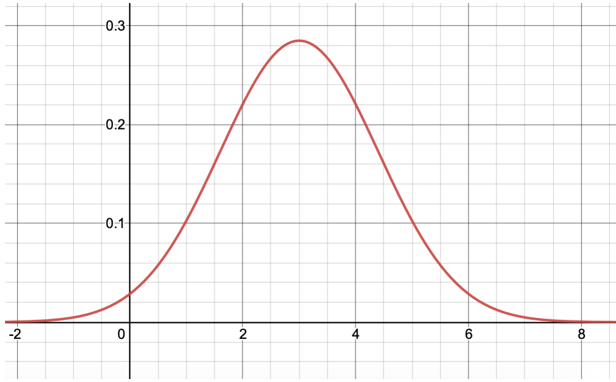
$$\log L(\theta; S) = \log P(S; \theta) = \log \mathrm{P}(\{x_i\}_{i=1}^{m}; \theta) = \log \prod_i P(x_i; \theta) = \sum_i \log P(x_i; \theta)$$

$$= \#\mathrm{H} \log P(x_i = \mathrm{H}; \theta) + \#\mathrm{T} \log P(x_i = \mathrm{T}; \theta) = \#\mathrm{H} \log \theta + \#\mathrm{T} \log(1 - \theta)$$

- What's the argmax? $\frac{d}{d\theta} \log L(\theta; S) = \frac{\#\mathrm{H}}{\theta} - \frac{\#\mathrm{T}}{1-\theta} = 0 \Rightarrow \hat{\theta} = \frac{\#\mathrm{H}}{\#\mathrm{H} + \#\mathrm{T}}$
- Think: is $\hat{\theta} = \theta^*$?

https://en.wikipedia.org/wiki/Coin_flipping

Example: Continuous variables



- Consider a dataset $S = \{x_i\}_{i=1}^{m}$, sampled from a Gaussian random variable $X \sim N(\mu, \sigma)$, where $\theta = (\mu, \sigma)$ [still no y in this example]
- What is $\hat{\theta} = \operatorname{argmax}_{\theta} \log L(\theta; S)$?

$$\log L(\theta; S) = \sum_i \log P(x_i; \theta) = \sum_i \log\left(\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)\right)$$

$$= -m \log(\sigma\sqrt{2\pi}) - \frac{1}{2\sigma^2} \sum_i (x_i - \mu)^2$$

$$\frac{d}{d\mu} \log L(\theta; S) = \frac{1}{\sigma^2} \sum_i (x_i - \mu) = 0 \quad \Rightarrow \quad \hat{\mu} = \frac{1}{m} \sum_i x_i$$

$$\frac{d}{d\sigma} \log L(\theta; S) = -\frac{m}{\sigma} + \frac{1}{\sigma^3} \sum_i (x_i - \mu)^2 = 0 \quad \Rightarrow \quad \hat{\sigma} = \sqrt{\frac{1}{m} \sum_i (x_i - \hat{\mu})^2}$$

Classification example: Naïve Bayes

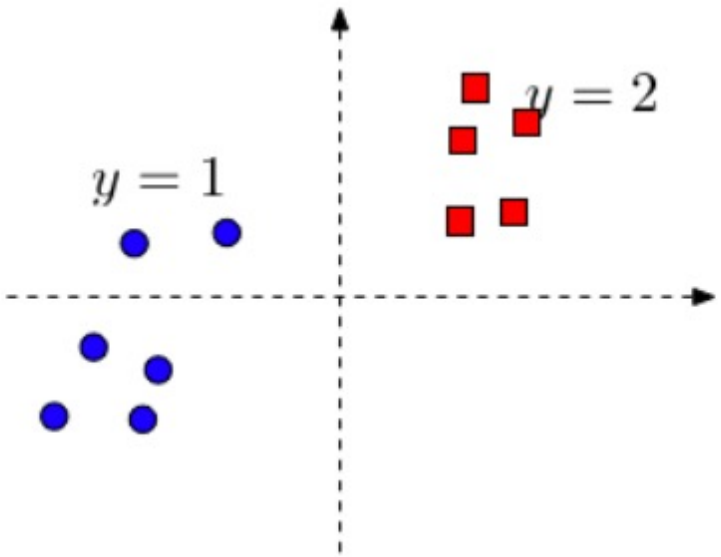
- Back to our **supervised setup**: $S = \{(x_i, y_i)\}_{i=1}^m$, and assume:
  - Each $x_i \in \mathcal{X} = \{0,1\}^d$ and $y_i \in \mathcal{Y} = \{0,1\}$
  - Denote $x_i[j]$ the j'th feature of the i'th example

Example datum:
$x = [0,1,1,0,0,1]$
$y = 1$

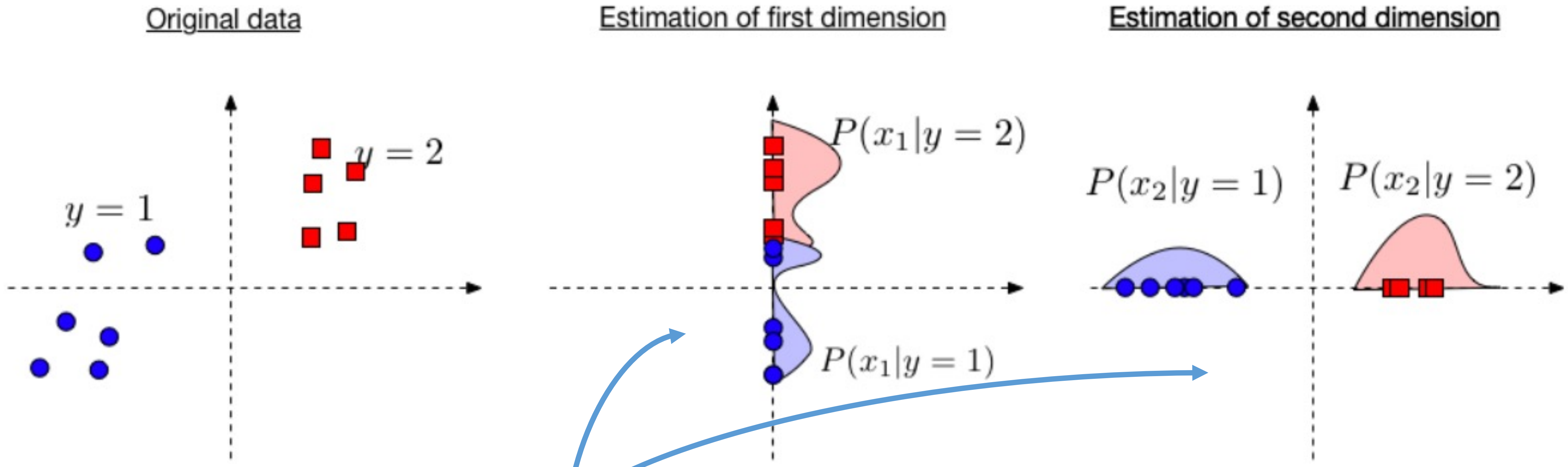
- Suppose we wanted to model $P(Y = y|X = x)$
- How many parameters would we need? ⟶ 2^d (think why)
- Naïve Bayes assumes that $P(x|y) = P(x[1], \dots, x[d]|y) = \prod_{j=1}^d P(x[j]|y)$
- The classifier: $h(x) = \text{argmax}_y P(y)P(x|y) = \text{argmax}_y P(y) \prod_{j=1}^d P(x[j]|y)$

⟶ Need to estimate only $2d + 1$ parameters! (less risk of overfitting)

Original data

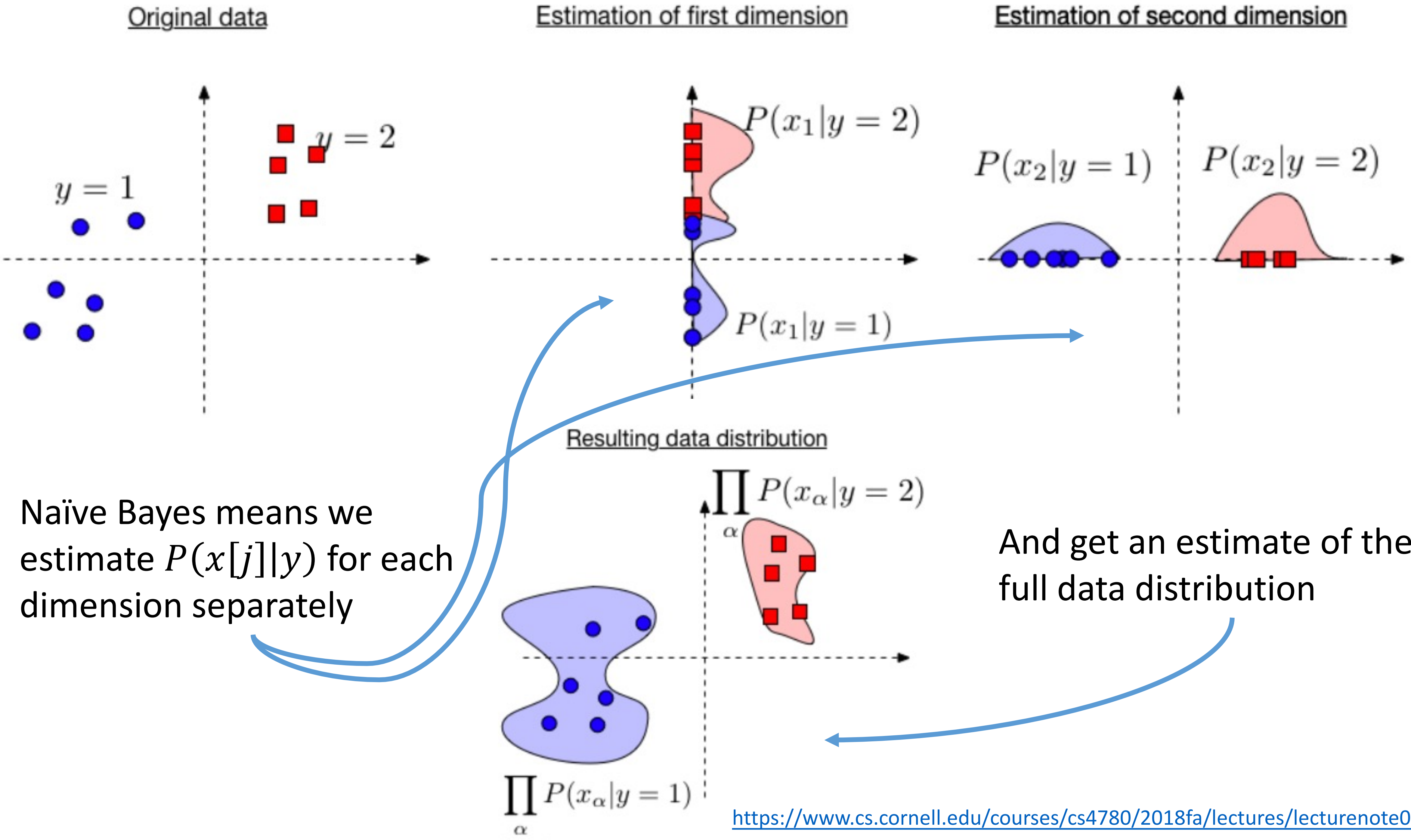


https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote05.html



Naïve Bayes means we estimate $P(x[j]|y)$ for each dimension separately

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote05.html



Naïve Bayes means we estimate $P(x[j]|y)$ for each dimension separately

And get an estimate of the full data distribution

https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote05.html

NB assumption:

$$P(x|y) = \prod_{j=1}^{d} P(x[j]|y)$$

Training Naïve Bayes

- How do we estimate $P(x|y;\theta)$ and $P(y;\theta)$?
- We've seen the log-likelihood:

$$\log L(\theta; S) = \sum_i \log P(x_i|y_i;\theta) + \sum_i \log P(y_i;\theta)$$

Training Naïve Bayes

NB assumption:

$$P(x|y) = \prod_{j=1}^{d} P(x[j]|y)$$

- How do we estimate $P(x|y;\theta)$ and $P(y;\theta)$?
- We've seen the log-likelihood:

$$\log L(\theta; S) = \sum_i \log P(x_i|y_i;\theta) + \sum_i \log P(y_i;\theta) = \sum_i \sum_j \log P(x_i[j]|y_i;\theta) + \sum_i \log P(y_i;\theta)$$

- What is $\hat{\theta} = \operatorname{argmax}_\theta \log L(\theta; S)$? What are the parameters θ?
- In the binary case ($x_i \in \{0,1\}^d$, $y_i \in \{0,1\}$):
  - $\theta_c^* = P(Y=1)$
  - $\theta_0^*[j] = P(X[j]=1|Y=0)$ (d terms)
  - $\theta_1^*[j] = P(X[j]=1|Y=1)$ (d terms)

MLE

Coin flips!

$$\hat{\theta}_c = \frac{\#\{Y=1\}}{m}$$

$$\hat{\theta}_0[j] = \frac{\#\{Y=0, X[j]=1\}}{\#\{Y=0\}}$$

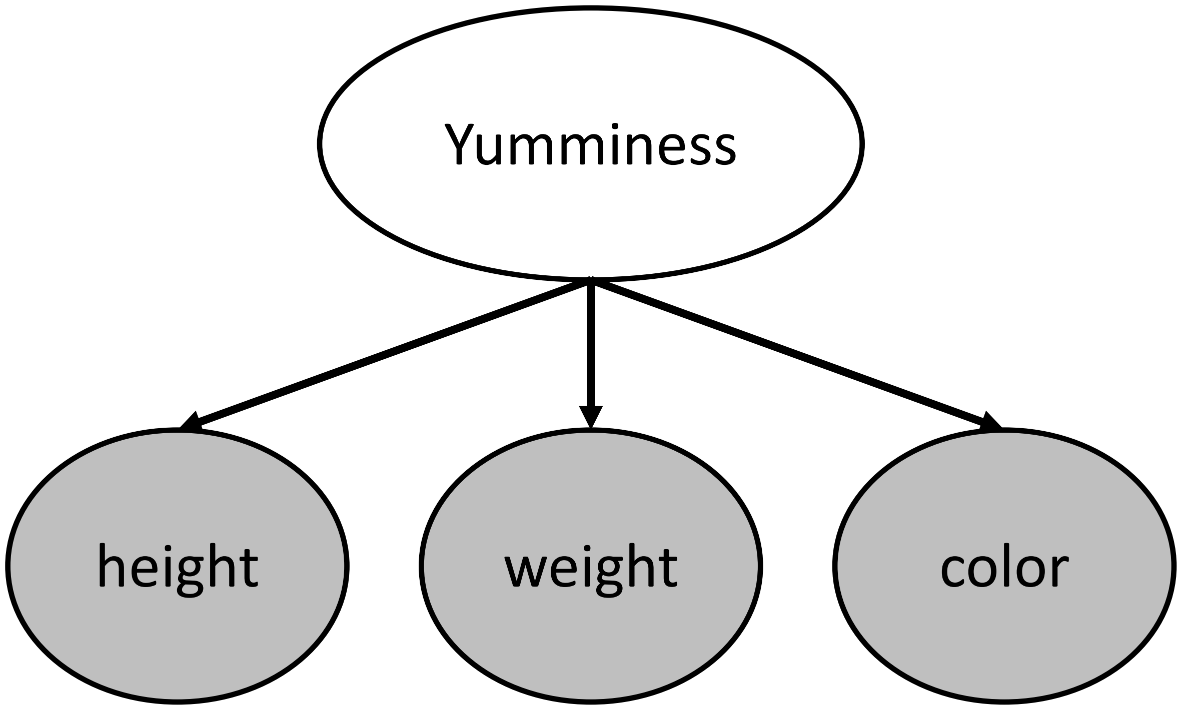
$$\hat{\theta}_1[j] = \frac{\#\{Y=1, X[j]=1\}}{\#\{Y=1\}}$$

Naïve Bayes: Discussion

NB assumption:

$$P(x|y) = \prod_{j=1}^{d} P(x[j]|y)$$

- In Naïve Bayes, the features are independent conditioned on the label
- **Generative story**:
  - First sample Yummy/Not
  - Then sample each feature idependently from the others
- Strong assumption; is it true?
- Probably not (heavy Papayas tend to be higher)
- But, can be useful



Naïve Bayes for text classification

- Consider a spam detection task: $\mathcal{Y} = \{\text{SPAM}, \text{NOTSPAM}\}$
- How should we represent a text T = "Click to win win"?
- Bag-of-words representation:

$$x_T = [0 \dots\ 0\ \dots\ \ 0\ 1\ 0 \dots 0\ 1\ 0 \dots 0\ 2\ 0 \dots 0] \in \mathbb{R}^{|V|}$$

  - $V = \{w_j\}_{j=1}^{|V|}$ is the vocabulary; $x[j]$ is number of times word w_j appears in T

- How to predict: $\text{argmax}_y P(y)P(x|y) =_{NB} \text{argmax}_y P(y) \underbrace{\prod_{i=1}^{|T|} P(T_i|y)}$

Probability to see word T_i in text with label y

$P(Click|y) * P(to|y)$
$* P(win|y) * P(win|y)$

- MLE: $P(\text{SPAM}) = \dfrac{\#\text{SPAM}}{m}$

$$P(w_j|\text{SPAM}) = \frac{\#\{w_j, \text{SPAM}\}}{\sum_w \#\{w, \text{SPAM}\}}$$

$$P(w_j|\text{NOTSPAM}) = \frac{\#\{w_j, \text{NOTSPAM}\}}{\sum_w \#\{w, \text{NOTSPAM}\}}$$

Naïve Bayes for text classification

- Consider a spam detection task: $\mathcal{Y} = \{\text{SPAM}, \text{NOTSPAM}\}$
- How should
- Bag-of-word

Think: what happens if we never saw the word “win” with SPAM docs during training? (We’ll get back to that)

$$x_T = [0 \dots 0 \;\; \dots \;\; 0\;1\;0 \dots 0\;1\;0 \dots 0\;2\;0 \dots 0] \in \mathbb{R}^{|V|}$$

  - $V = \{w_j\}_{j=1}^{|V|}$ is the vocabulary; $x[j]$ is number of times word w_j appears in T

- How to predict: $\text{argmax}_y P(y)P(x|y) =_{NB} \text{argmax}_y P(y) \underbrace{\prod_{i=1}^{|T|} P(T_i|y)}_{P(Click|y) * P(to|y) * P(win|y) * P(win|y)}$

Probability to see word T_i in text with label y

- MLE: $P(\text{SPAM}) = \dfrac{\#\text{SPAM}}{m}$

$$P(w_j|\text{SPAM}) = \frac{\#\{w_j, \text{SPAM}\}}{\sum_w \#\{w, \text{SPAM}\}}$$

$$P(w_j|\text{NOTSPAM}) = \frac{\#\{w_j, \text{NOTSPAM}\}}{\sum_w \#\{w, \text{NOTSPAM}\}}$$

Naïve Bayes is (sometimes) a linear classifier

- Binary case: $x_i \in \mathcal{X} = \{0,1\}^d$ and $y_i \in \mathcal{Y} = \{0,1\}$
- We predict 1 if $P(Y = 1|x) > P(Y = 0|x)$

Naïve Bayes is (sometimes) a linear classifier

- Binary case: $x_i \in \mathcal{X} = \{0,1\}^d$ and $y_i \in \mathcal{Y} = \{0,1\}$
- We predict 1 if $P(Y = 1|x) > P(Y = 0|x)$

$$\Leftrightarrow \frac{P(x|Y = 1)P(Y = 1)}{P(x|Y = 0)P(Y = 0)} > 1$$

Naïve Bayes is (sometimes) a linear classifier

- Binary case: $x_i \in \mathcal{X} = \{0,1\}^d$ and $y_i \in \mathcal{Y} = \{0,1\}$
- We predict 1 if $P(Y = 1|x) > P(Y = 0|x)$

$$\Leftrightarrow \frac{P(x|Y=1)P(Y=1)}{P(x|Y=0)P(Y=0)} > 1 \Leftrightarrow \frac{P(Y=1)}{P(Y=0)} \prod_{j=1}^{d} \frac{P(x[j]|Y=1)}{P(x[j]|Y=0)} > 1 \qquad (*)$$

Naïve Bayes is (sometimes) a linear classifier

- Binary case: $x_i \in \mathcal{X} = \{0,1\}^d$ and $y_i \in \mathcal{Y} = \{0,1\}$
- We predict 1 if $P(Y = 1|x) > P(Y = 0|x)$

$$\Leftrightarrow \frac{P(x|Y=1)P(Y=1)}{P(x|Y=0)P(Y=0)} > 1 \Leftrightarrow \frac{\overset{p}{P(Y=1)}}{\underset{1-p}{P(Y=0)}} \prod_{j=1}^{d} \frac{\overbrace{P(x[j]|Y=1)}^{a_j}}{\underbrace{P(x[j]|Y=0)}_{b_j}} > 1 \qquad (*)$$

  - Then $P(x[j]|Y=1) = a_j^{x[j]}(1-a_j)^{(1-x[j])}$ and $P(x[j]|Y=0) = b_j^{x[j]}(1-b_j)^{(1-x[j])}$
- (*) becomes $\frac{p}{1-p} \prod_{j=1}^{d} \frac{a_j^{x[j]}(1-a_j)^{(1-x[j])}}{b_j^{x[j]}(1-b_j)^{(1-x[j])}} > 1$

Naïve Bayes is (sometimes) a linear classifier

$$\frac{p}{1-p}\prod_{j=1}^{d}\frac{a_j^{x[j]}\left(1-a_j\right)^{(1-x[j])}}{b_j^{x[j]}\left(1-b_j\right)^{(1-x[j])}} > 1$$

Rearrange terms

$$\Leftrightarrow \frac{p}{1-p}\prod_{j=1}^{d}\frac{1-a_j}{1-b_j}\prod_{j=1}^{d}\left(\frac{a_j\left(1-b_j\right)}{b_j\left(1-a_j\right)}\right)^{x_j} > 1$$

Take log

$$\Leftrightarrow \underbrace{\log\frac{p}{1-p}\prod_{j=1}^{d}\frac{1-a_j}{1-b_j}}_{b} + \sum_{j=1}^{d} x_j \underbrace{\log\left(\frac{a_j\left(1-b_j\right)}{b_j\left(1-a_j\right)}\right)}_{\boxed{w_j}} > 0$$

$$\Leftrightarrow b + \sum_{j}^{d} x_j w_j > 0 \Leftrightarrow b + w^T x > 0$$

Linear model!

Naïve Bayes: Gaussian case

- Assume that $\mathrm{X}[j]|Y = y \sim N(\mu_y[j], \sigma[j])$, such that $x_i \in \mathcal{X} = \mathbb{R}^d$ and $y_i \in \mathcal{Y} = \{0,1\}$ [note $\sigma[j]$ doesn't depend on y here]
- What are the parameters θ to estimate?
  - $\theta_c^* = P(Y = 1)$
  - $\mu_y[j]$, for all y, j ($2d$ terms)
  - $\sigma[j]$, for all j (d terms)
- MLE:

$$\log L(\theta; S) = \sum_i \sum_j \log P(x_i[j] | y_i; \theta) + \sum_i \log P(y_i; \theta)$$

MLE

$$\hat{\theta}_c = \frac{\#\{Y = 1\}}{m}$$

$$\hat{\mu}_y[j] = \frac{1}{\#\{i: y_i = y\}} \sum_{i: y_i = y} x_i[j]$$

$$\hat{\sigma}[j] = \sqrt{\frac{1}{m} \sum_i \left(x_i[j] - \hat{\mu}_{y_i}[j]\right)^2}$$

Naïve Bayes is (sometimes) a linear classifier

- We saw that binary Naïve Bayes is linear
- What about Gaussian Naïve Bayes?
- It's possible to show that in this case:
- $P(Y=1|x) = \frac{1}{1+e^{w^T x+b}}$, where:

$$w[j] = \frac{\mu_0[j] - \mu_1[j]}{\sigma[j]^2}$$

$$b = \log \frac{1-\theta_c^*}{\theta_c^*} + \sum_j \frac{\mu_1[j]^2 - \mu_0[j]^2}{2\sigma[j]^2}$$

- Does that look familiar?
- Same form as logistic regression!
- **Think**: which parameters do we estimate in each case, and how?

Naïve Bayes is (sometimes) a linear classifier

- Moreover, we predict 1 if $P(Y = 1|x) > P(Y = 0|x)$

$$\Leftrightarrow \frac{1}{1+e^{w^T x+b}} / \frac{e^{w^T x+b}}{1+e^{w^T x+b}} > 1$$

Naïve Bayes is (sometimes) a linear classifier

- Moreover, we predict 1 if $P(Y = 1|x) > P(Y = 0|x)$

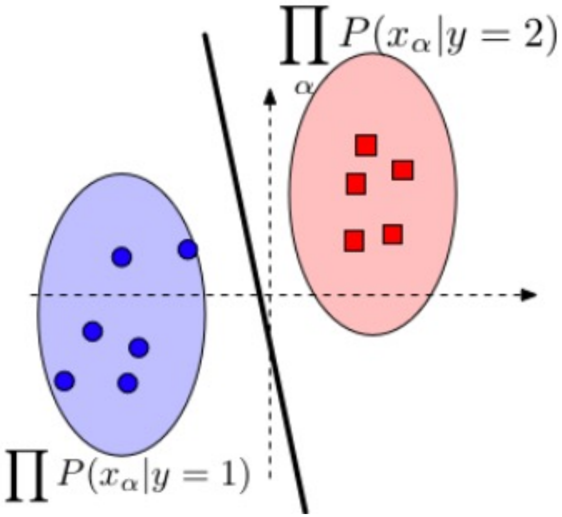
$$\Leftrightarrow \frac{1}{1+e^{w^T x+b}} / \frac{e^{w^T x+b}}{1+e^{w^T x+b}} > 1 \Leftrightarrow e^{w^T x+b} < 1$$

Naïve Bayes is (sometimes) a linear classifier

- Moreover, we predict 1 if $P(Y = 1|x) > P(Y = 0|x)$

$$\Leftrightarrow \frac{1}{1+e^{w^T x+b}} / \frac{e^{w^T x+b}}{1+e^{w^T x+b}} > 1 \Leftrightarrow e^{w^T x+b} < 1 \Leftrightarrow w^T x + b < 0$$

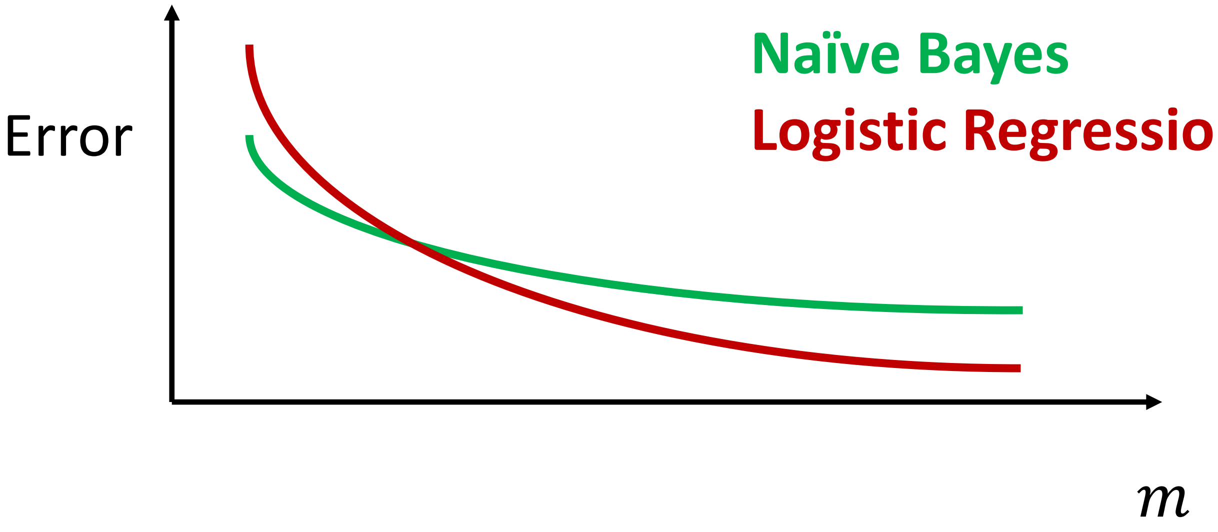
- Gaussian Naïve Bayes (and logistic regression) are linear!



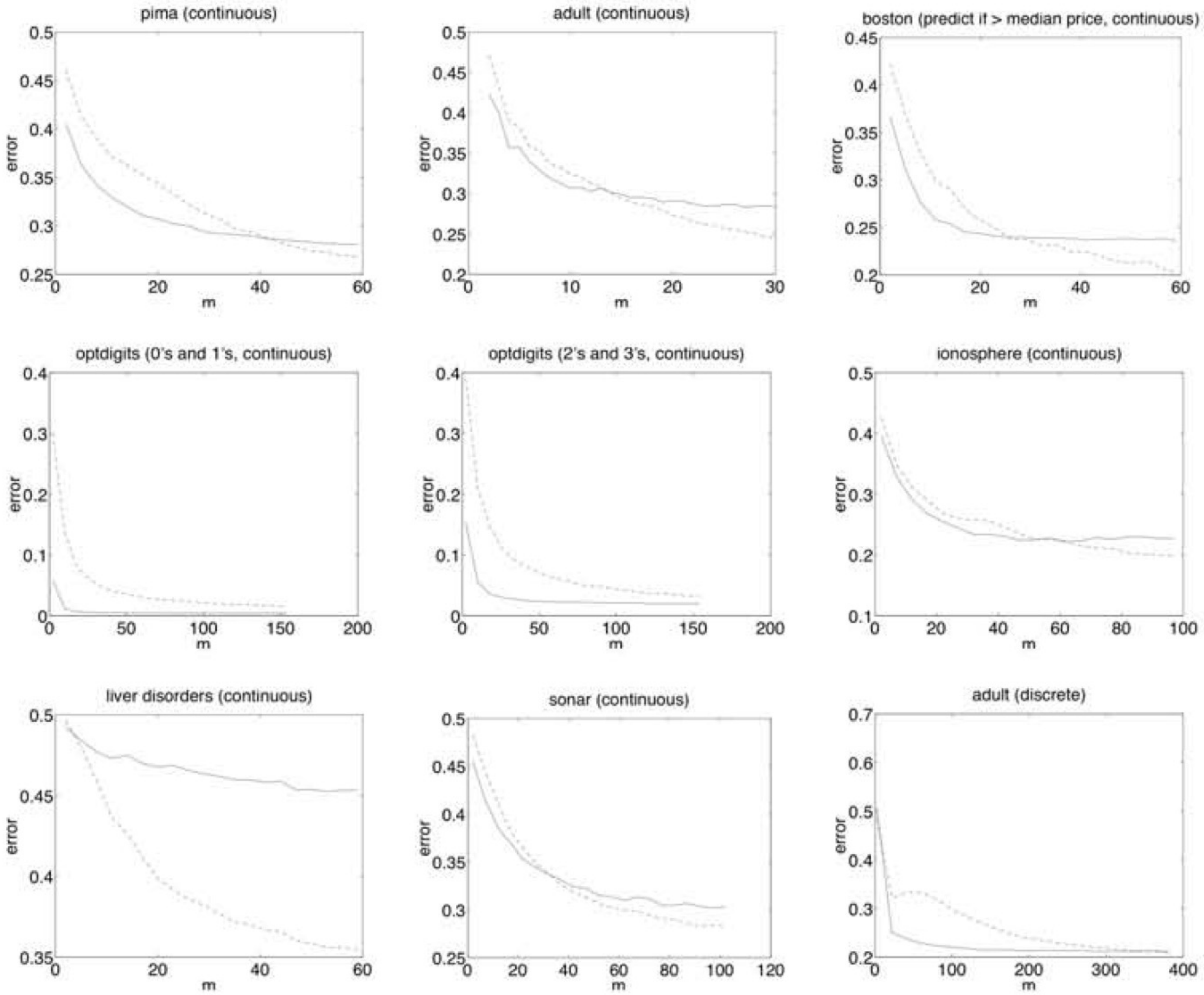
https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote05.html

	Logistic Regression		Naïve Bayes
Model Family	Discriminative $P(y\|x)$		Generative $P(x\|y)$
Error	$L_D(h_S)$	$\leq$	$L_D(h_S)$
Sample complexity	$m = O(d)$		$m = O(\log d)$

Question: How should the following plot look like?



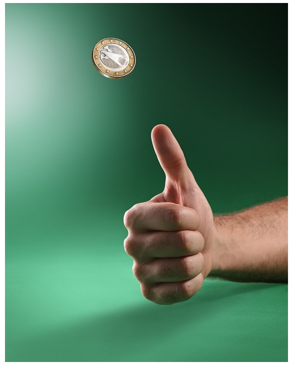
[Ng&Jordan 2000]



[Ng&Jordan 2000]

Bayesian reasoning and MAP estimation

Example: Biased coin



- Suppose you see the following series of coin flips: T, T, T
- What is the MLE estimate of $\theta = P(H)$? $\rightarrow \frac{\#H}{m} = \frac{0}{3} = 0$
- Is this good or bad?
- Recall $\log L(\theta; S) = \#\mathrm{H} \log \theta + \#\mathrm{T} \log(1-\theta)$ $\rightarrow L(\theta; S) = -\infty$
- Overfitting: if θ is small, with probability $(1-\theta)^m$ will see all T
- How can we fix this?
- Idea: add pseudo-examples, pairs of H and T: H, T, T, T, T
- What's the estimate of $\theta = P(H)$ now? $\rightarrow \frac{\#H}{m} = \frac{1}{5} = \frac{0+1}{3+2}$

https://en.wikipedia.org/wiki/Coin_flipping

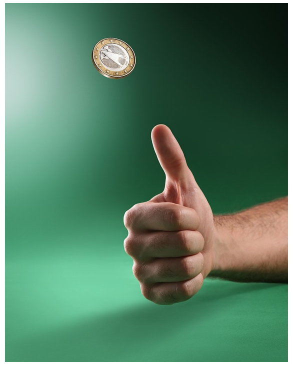
Bayesian reasoning

- So far we thought of θ as unkown (but fixed!) parameters that we estimate with MLE
- However, this approach can overfit, especially with little data
- In Bayesian reasoning, we think of θ as a random variable
- $P(\theta)$ is the prior distribution
  - Chosen by us before learning – a modeling decision
  - Means that samples $\{x_i\}$ are now independent only conditioned on θ

Maximum a posteriori (MAP) Estimation

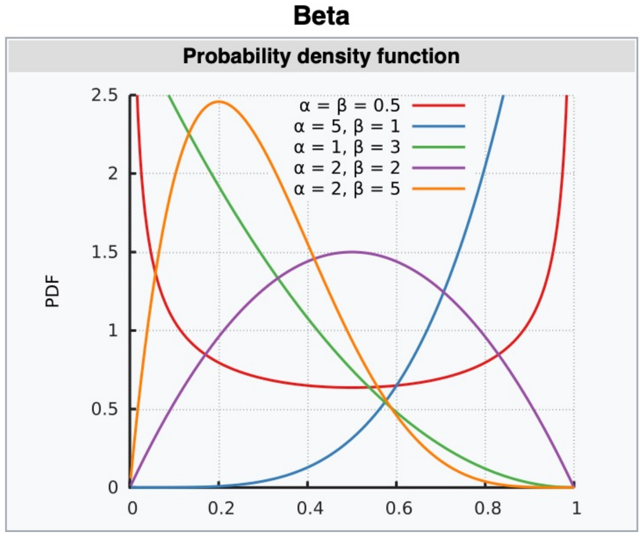
- In MLE, we look for $\mathrm{argmax}_\theta P(S|\theta)$
- In MAP, we look for $\mathrm{argmax}_\theta \underbrace{P(\theta|S)}_{\text{Posterior}} = \mathrm{argmax}_\theta \underbrace{P(S|\theta)}_{\text{Likelihood}} \underbrace{P(\theta)}_{\text{Prior}}$
- Consequences
  - With uniform prior, MAP is the same as MLE
  - As we get more data, the MAP estimate converges to the MLE one
- How do we perform MAP?
  - First, assume a prior
  - Then, optimize

Example: Biased coin

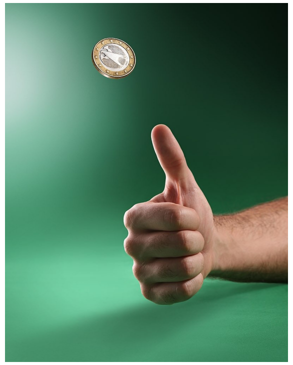


- Given a sequence of coin flips $S = \{x_1, \dots, x_m\}$
- Likelihood $P(S|\theta) = \prod_i \theta^{x_i}(1-\theta)^{1-x_i} = \theta^{\#H}(1-\theta)^{\#T}$
- Let's assume prior $P(\theta) = Beta(\theta|\alpha,\beta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$
- Posterior: $P(\theta|S) \propto P(S|\theta)P(\theta) = \theta^{\alpha-1+\#H}(1-\theta)^{\beta-1+\#T}$
- We get $P(\theta|S) = Beta(\theta|\alpha+\#H, \beta+\#T)$
  (Beta is conjugate prior for the Bernoulli likelihood)
- Differentiating w.r.t θ and equating to 0, we get:

$$\hat{\theta}_{MAP} = \frac{\alpha+\#H-1}{\alpha+\beta+\#H+\#T-2}$$

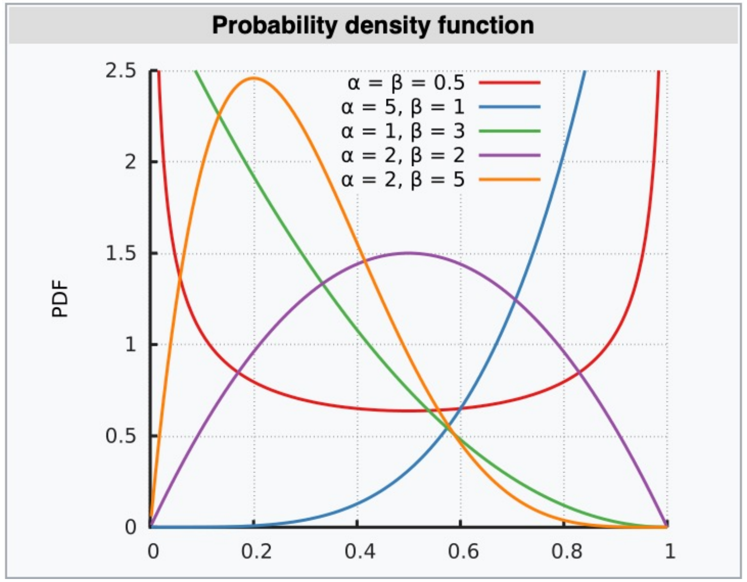


Example: Biased coin



- We got $\hat{\theta}_{MAP} = \frac{\alpha+\#H-1}{\alpha+\beta+\#H+\#T-2}$
- Let's set $\alpha = 1, \beta = 1$
  - Then $P(\theta) = Beta(\theta|\alpha,\beta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1} = 1$ is a uniform prior
  - And $\hat{\theta}_{MAP} = \frac{1+\#H-1}{1+1+\#H+\#T-2} = \frac{\#H}{\#H+\#T} = \hat{\theta}_{MLE}$
- Now let's set $\alpha = 2, \beta = 2$
  - Then $\hat{\theta}_{MAP} = \frac{2+\#H-1}{2+2+\#H+\#T-2} = \frac{\#H+1}{\#H+\#T+2}$ which is our estimate using pseudo-examples
  - Prior $Beta(\theta|\alpha,\beta)$ is like adding $(\alpha-1)$ H and $(\beta-1)$ T pseudo-examples

Beta



Example: Binary Naïve Bayes

NB assumption:

$$P(x|y) = \prod_{j=1}^{d} P(x[j]|y)$$

- Recall that for the binary case ($x_i \in \{0,1\}^d$, $y_i \in \{0,1\}$):
  - $\theta_c^* = P(Y = 1)$
  - $\theta_0^*[j] = P(X[j] = 1|Y = 0)$ (d terms)
  - $\theta_1^*[j] = P(X[j] = 1|Y = 1)$ (d terms)

MLE

Coin flips!

$$\hat{\theta}_c = \frac{\#\{Y = 1\}}{m}$$

$$\hat{\theta}_0[j] = \frac{\#\{Y = 0, X[j] = 1\}}{\#\{Y = 0\}}$$

$$\hat{\theta}_1[j] = \frac{\#\{Y = 1, X[j] = 1\}}{\#\{Y = 1\}}$$

- What are the MAP estimates?
  - Need to specify **priors** (for all params)
  - E.g., Beta with $\alpha = 2, \beta = 2$
  - Smoothing the zero probabilities of unseen events

MAP

Coin flips!

$$\hat{\theta}_c = \frac{\#\{Y = 1\} + 1}{m + 2}$$

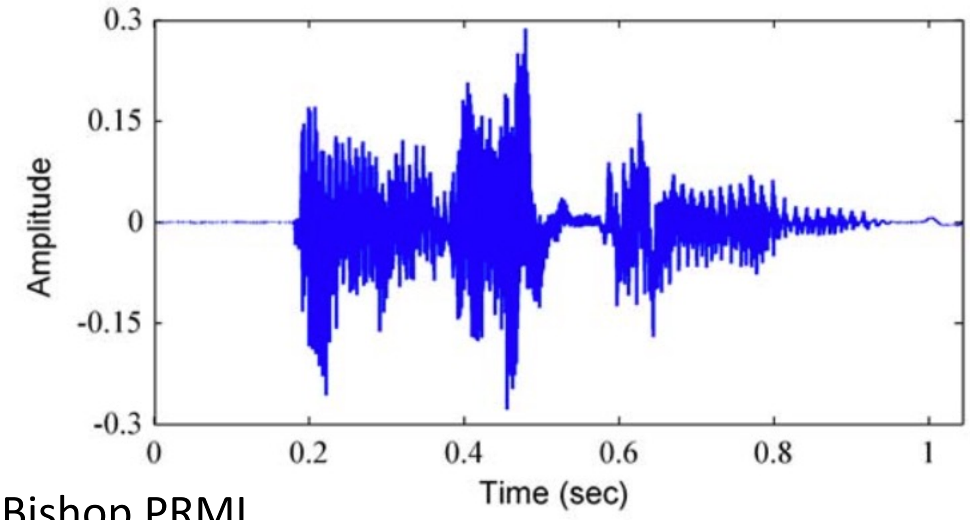
$$\hat{\theta}_0[j] = \frac{\#\{Y = 0, X[j] = 1\} + 1}{\#\{Y = 0\} + 2}$$

$$\hat{\theta}_1[j] = \frac{\#\{Y = 1, X[j] = 1\} + 1}{\#\{Y = 1\} + 2}$$

Other generative models

Sequential data

- So far, we've considered "simple" classification setups:
  - Binary labels $y_i \in \{0,1\}$
  - Vector or disccrete inputs, e.g., $x_i \in \{0,1\}^d$
- But real-world data often look different
  - Many problems are sequential

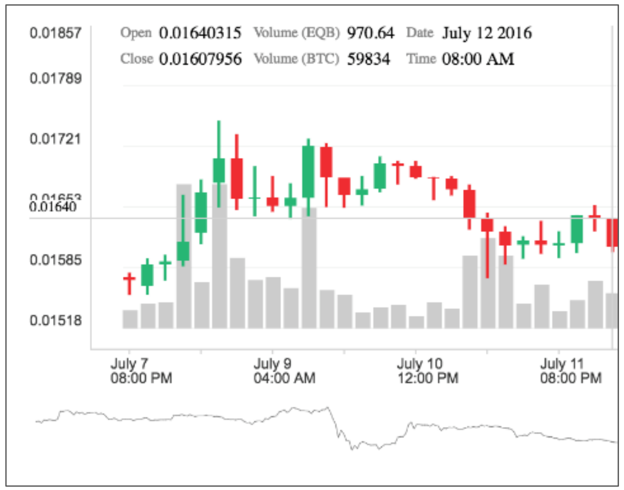


Bishop PRML

Goodbye and thanks for all the fish

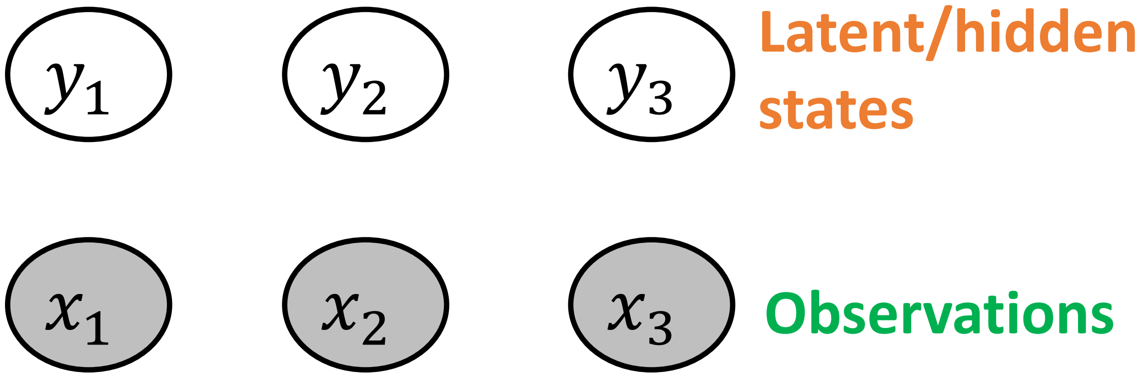
وداعا وشكرا لجميع الأسماك

Google Translate



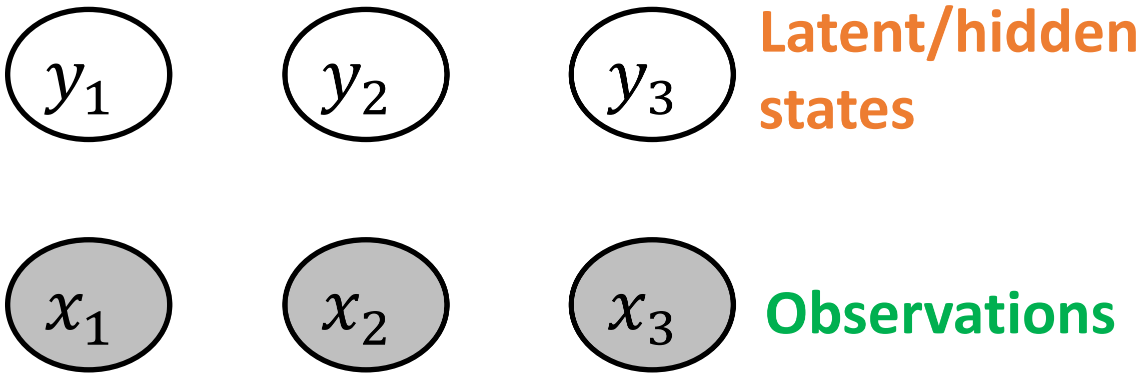
Sequential data

- Inputs $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{m}$, where
  $x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \ldots, x_{T_i}^{(i)}\}$, $x_k^{(i)}$ may be discrete or continuous
  $y^{(i)} = \{y_1^{(i)}, y_2^{(i)}, \ldots, y_{T_i}^{(i)}\}$, $y_k^{(i)} \in \{0,1\}$



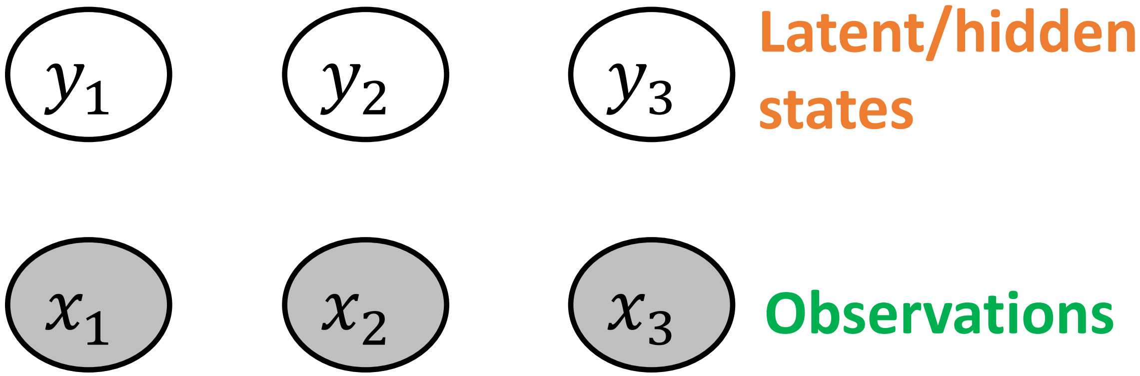
Sequential data

- Inputs $S = \{x^{(i)}, y^{(i)}\}_{i=1}^{m}$, where
  $x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \dots, x_{T_i}^{(i)}\}$, $x_k^{(i)}$ may be discrete or continuous
  $y^{(i)} = \{y_1^{(i)}, y_2^{(i)}, \dots, y_{T_i}^{(i)}\}$, $y_k^{(i)} \in \{0,1\}$
- **Example**: Ice cream and weather (Eisner 2002)
  - Observations: number of ice creams eaten each day: $\mathcal{X} = \{1,2,3\}$
  - Hidden variables:
    Weather on each day:
    cold (C) or hot (H), so $\mathcal{Y} = \{C, H\}$



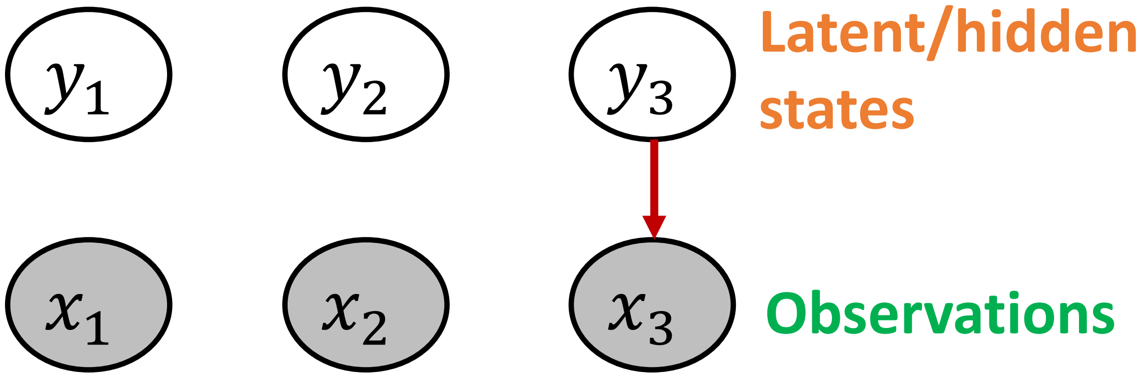
Sequential data

- **Example**: Ice cream and weather
  - Observations: number of ice creams eaten each day: $\mathcal{X} = \{1,2,3\}$
  - Hidden variables: Weather on each day: $\mathcal{Y} = \{C, H\}$
- Suppose we want to model the probability of eating 2 ice creams on day 3. What could it depend on?
- $P(X_3 = 2 | ?)$



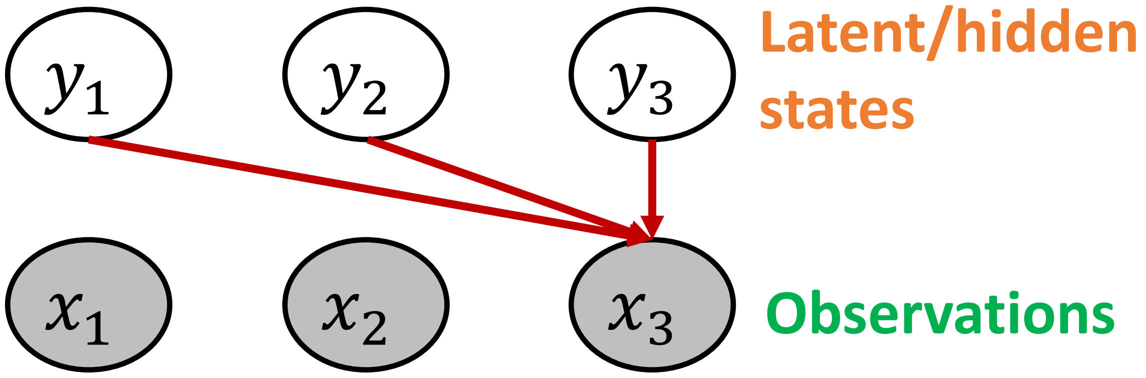
Sequential data

- **Example**: Ice cream and weather
  - Observations: number of ice creams eaten each day: $\mathcal{X} = \{1,2,3\}$
  - Hidden variables: Weather on each day: $\mathcal{Y} = \{C, H\}$
- Suppose we want to model the probability of eating 2 ice creams on day 3. What could it depend on?
- $P(X_3 = 2|y_3)$ – Only the weather at day 3



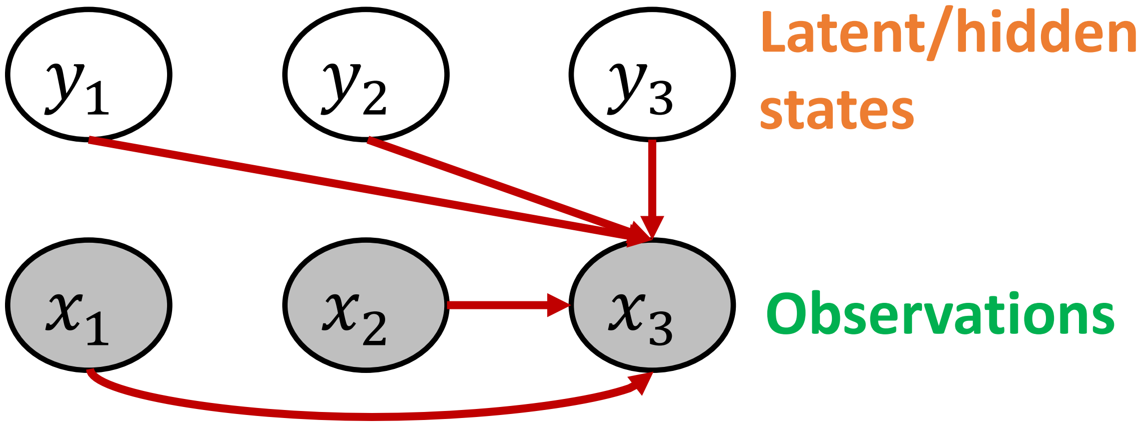
Sequential data

- **Example**: Ice cream and weather
  - Observations: number of ice creams eaten each day: $\mathcal{X} = \{1,2,3\}$
  - Hidden variables: Weather on each day: $\mathcal{Y} = \{C, H\}$
- Suppose we want to model the probability of eating 2 ice creams on day 3. What could it depend on?
- $P(X_3 = 2 | y_1, y_2, y_3)$ – Also weather on other days



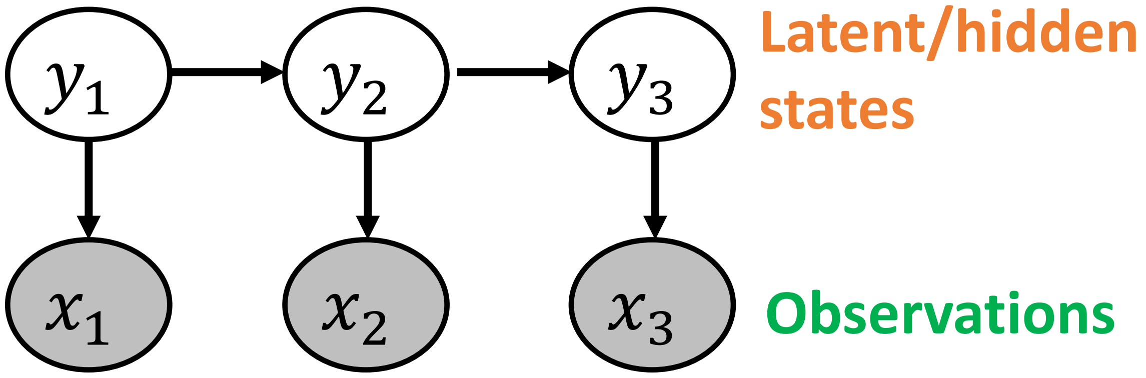
Sequential data

- **Example**: Ice cream and weather
  - Observations: number of ice creams eaten each day: $\mathcal{X} = \{1,2,3\}$
  - Hidden variables: Weather on each day: $\mathcal{Y} = \{C, H\}$
- Suppose we want to model the probability of eating 2 ice creams on day 3. What could it depend on?
- $P(X_3 = 2 | y_1, y_2, y_3, x_1, x_2)$ – Also previous # of ice creams
- Pros/cons of different options



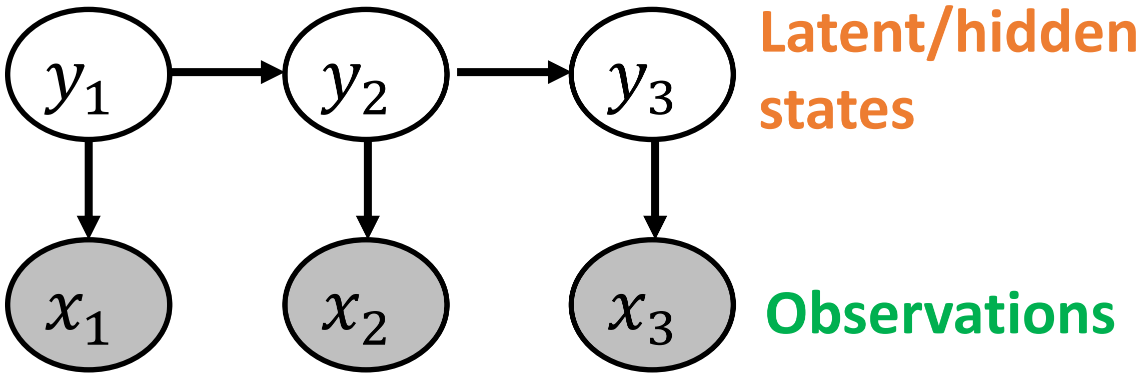
Hidden Markov Model (HMM)

- The HMM imposes a specific dependence structure:
  - Each observation depends only on its corresponding latent state
  - Each latent state depends only on its previous one
- **Generative story**:
  - First sample a sequence of hidden states
  - For each hidden state, sample the features
- Strong assumption; is it true?
  - Probably not
  - But, can be useful



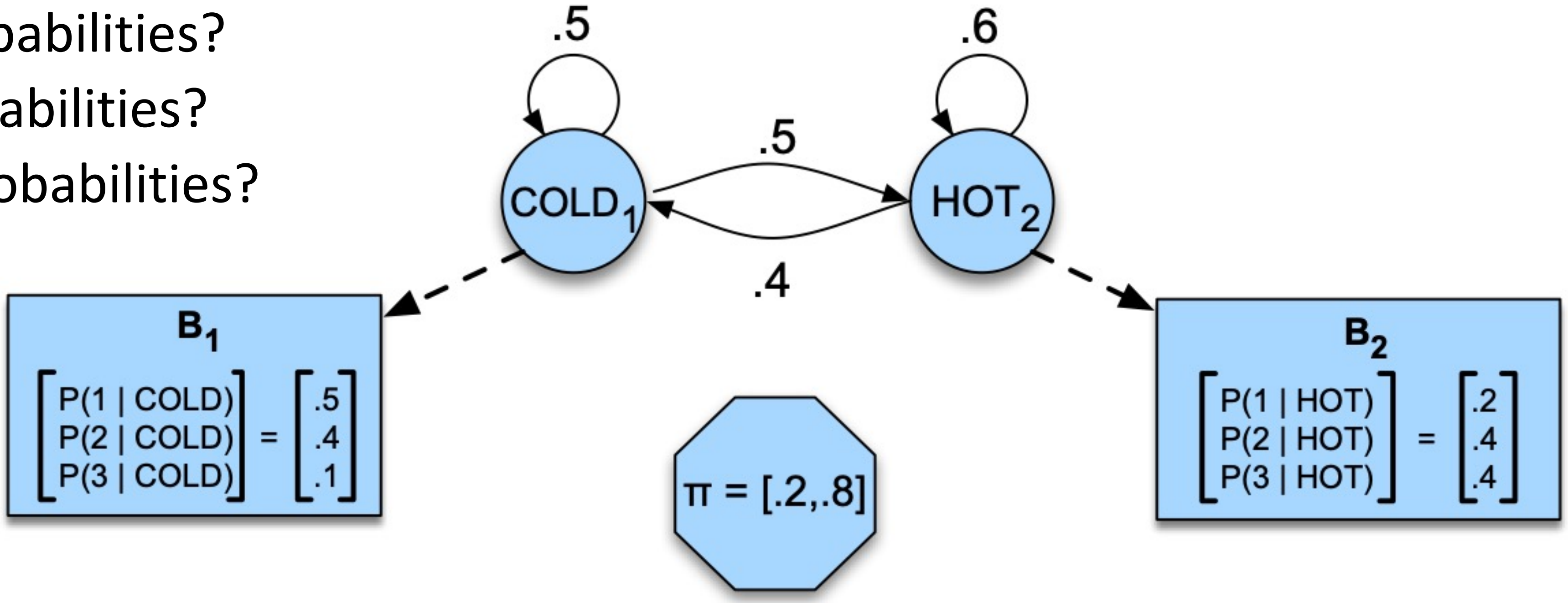
Hidden Markov Model (HMM)

- An HMM is specified by:
  - A set of states $\mathcal{Y}$, e.g., $\mathcal{Y} = \{0,1\}$
  - Transition probabilities θ: $P(Y_{t+1} = y | Y_t = y') \quad \forall y, y'$
  - A sequence of observations $x_1, \ldots, x_T$
  - Emission probabilities ϕ: $P(x|y) \quad \forall x, y$
  - Initial state probabilties π_y – probability of starting at state y, $\forall y$



Example: Ice cream task (Eisner 2002)

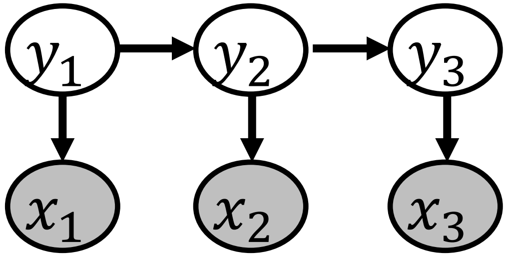
- **Example**: Ice cream and weather
  - Observations: number of ice creams eaten each day: $\mathcal{X} = \{1,2,3\}$
  - Hidden variables: Weather on each day: cold (C) or hot (H), so $\mathcal{Y} = \{C, H\}$
- Example HMM (state machine) – what are
  - the transition propabilities?
  - the emission probabilities?
  - the initial state probabilities?



The three problems of HMM

1. **Likelihood**: Given an HMM with parameters (θ, ϕ, π) and an observation sequence $\mathbf{x} = x_1, \ldots, x_T$, determine the likelihood $P(\mathbf{x}; \theta, \phi, \pi)$
2. **Decoding**: Given an observation sequence $\mathbf{x} = x_1, \ldots, x_T$ and an HMM, find the most probable hidden state sequence $\mathbf{y} = y_1, \ldots, y_T$
3. **Learning**: Given an observation sequence $\mathbf{x} = x_1, \ldots, x_T$ and the set of HMM states $\mathcal{Y}$, learn the HMM parameters (θ, ϕ, π)

Likelihood computation



- Given an HMM with parameters (θ, ϕ, π) and an observation sequence $\mathbf{x} = x_1, \dots, x_T$, determine the likelihood $P(\mathbf{x}; \theta, \phi, \pi)$
- **Problem**: we don't know the state sequence $\mathbf{y} = y_1, \dots, y_T$
- **Solution**: marginalize over it:

$$P(\mathbf{x}; \theta, \phi, \pi) = \sum_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{y}} P(\mathbf{x}|\mathbf{y}) P(\mathbf{y})$$

- How many terms in the sum? $|\mathcal{Y}|^T$
- The forward algorithm:
  dynamic program that runs in $|\mathcal{Y}|^2 T$

$$P(\mathbf{x}|\mathbf{y}) = \prod_{t=1}^{T} P(x_t|y_t)$$

$$P(\mathbf{y}) = P(\pi_{y_1}) \prod_{t=1}^{T-1} P(y_{t+1}|y_t)$$

Decoding

- Given an observation sequence $\mathbf{x} = x_1, \dots, x_T$ and an HMM, find the most probable hidden state sequence $\mathbf{y} = y_1, \dots, y_T$
- Could we just compute $P(\mathbf{x}|\mathbf{y})$ for each $\mathbf{y}$ and take argmax?
- No! There are exponentially many $\mathbf{y}$'s
- The Viterbi algorithm: another dynamic program with $|\mathcal{Y}|^2 T$ runtime

Learning

- Given an observation sequence $\mathbf{x} = x_1, \ldots, x_T$ and the set of HMM states $\mathcal{Y}$, learn the HMM parameters (θ, ϕ, π)
- Note: this is a not a supervised learning problem!
  - We always assume we have x's and y's and need to learn parameters
  - Here we don't have y's
- There are ways to solve it, primarily the forward-backward algorithm (aka Baum-Welch, a special case of Expectation-Maximization)
- But this is beyond our scope

MLE for HMMs

- Inputs $S = \left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{m}$, where
  $x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \ldots, x_{T_i}^{(i)}\}, \qquad x_k^{(i)} \in \{0,1\}$
  $y^{(i)} = \{y_1^{(i)}, y_2^{(i)}, \ldots, y_{T_i}^{(i)}\}, \qquad y_k^{(i)} \in \{0,1\}$
- How do we estimate the parameters (θ, ϕ, π)?
  - $\pi_0 = \frac{\#\{y_1^{(i)}=0\}}{m}$
  - $P(Y_{t+1} = 0 | Y_t = 0) = \frac{\#\{Y_{t+1}=0, Y_t=0\}}{\#\{Y_t=0\}}$, $P(Y_{t+1} = 0 | Y_t = 1) = \frac{\#\{Y_{t+1}=0, Y_t=1\}}{\#\{Y_t=1\}}$
  - $P(X_t = 0 | Y_t = 0) = \frac{\#\{X_t=0, Y_t=0\}}{\#\{Y_t=0\}}$, $P(X_t = 0 | Y_t = 1) = \frac{\#\{X_t=0, Y_t=1\}}{\#\{Y_t=1\}}$

Example: MLE for HMM

- Back to the ice cream and weather example
- Imagine we see the following dataset:

3	3	2	1	1	2	1	2	3
hot	hot	cold	cold	cold	cold	cold	hot	hot

- $\pi_H =$

Example: MLE for HMM

- Back to the ice cream and weather example
- Imagine we see the following dataset:

3	3	2	1	1	2	1	2	3
hot	hot	cold	cold	cold	cold	cold	hot	hot

- $\pi_H = 1/3, \ \ \pi_C = 2/3$
- θ: $P(H|H) =$

Example: MLE for HMM

- Back to the ice cream and weather example
- Imagine we see the following dataset:

3	3	2	1	1	2	1	2	3
hot	hot	cold	cold	cold	cold	cold	hot	hot

- $\pi_H = 1/3, \ \ \pi_C = 2/3$
- θ: $P(H|H) = 2/3, \ \ P(C|H) = 1/3, \ \ P(C|C) = 2/3, \ \ P(H|C) = 1/3$
- ϕ: $P(1|H) =$

Example: MLE for HMM

- Back to the ice cream and weather example
- Imagine we see the following dataset:

3	3	2	1	1	2	1	2	3
hot	hot	cold	cold	cold	cold	cold	hot	hot

- $\pi_H = 1/3, \;\; \pi_C = 2/3$
- θ: $P(H|H) = 2/3, \;\; P(C|H) = 1/3, \;\; P(C|C) = 2/3, \;\; P(H|C) = 1/3$
- ϕ: $P(1|H) = 0/4, \;\; P(2|H) = 1/4, \;\; P(3|H) = 3/4$
  $P(1|C) = 3/5, \;\; P(2|C) = 2/5, \;\; P(3|C) = 0/5$

Discussion

- Generative vs discriminative models
  - Generative models estimate $P(x|y)$ and $P(y)$
  - Discriminative models estimate $P(y|x)$ directly
- **Heuristically**, generative models may be useful when
  - Want to impose some parametric distributions
  - Don’t have a lot of data
  - Want to generate data
    - Did not show
    - Lots of work (Dalle-2, chatGPT, etc. are related, thought not the same)
- Beware: these are heuristics!

Discussion

- Parameter estimation with MLE or MAP
  - MLE when have enough data
  - MAP when have little data and/or want to assume prior on θ
- Naïve Bayes
  - Assumes features are independent conditioned on label
  - Naïve, but often useful
  - Linear model, connection with logistic regression
- HMM
  - Assumes observations only depend on current state + Markov assumption
  - (Used to be) useful in many applications

Next week(s)

- **part III**: *more supervised learning*
    1. ~~Regression~~
    2. ~~Bagging and Boosting~~
    3. Generative models
    4. Deep learning