



Xi'an Jiaotong-Liverpool University

西交利物浦大学

**INT305 Machine Learning**

**Lecture 9**

**Probabilistic Models**

**Siyue Yu**

**Department Intelligence Science**

**[siyue.yu02@xjtlu.edu.cn](mailto:siyue.yu02@xjtlu.edu.cn)**

# Today

---

- So far in the course we have adopted a modular perspective, in which the model, loss function, optimizer, and regularizer are specified separately.
- Today we will begin putting together a **probabilistic interpretation** of the choice of model and loss, and introduce the concept of **maximum likelihood estimation**.
- Let's start with a simple biased coin example.
  - You flip a coin  $N = 100$  times and get outcomes  $\{x_1, \dots, x_N\}$  where  $x_i \in \{0, 1\}$  and  $x_i = 1$  is interpreted as heads  $H$ .
  - Suppose you had  $N_H = 55$  heads and  $N_T = 45$  tails.
  - What is the probability it will come up heads if we flip again? Let's design a model for this scenario, fit the model. We can use the fit model to predict the next outcome.

# Model?

---

- The coin is possibly loaded. So, we can assume that one coin flip outcome  $x$  is a **Bernoulli random variable** for *some currently unknown parameter*  $\theta \in [0, 1]$ .

$$p(x = 1|\theta) = \theta \quad \text{and} \quad p(x = 0|\theta) = 1 - \theta$$

or more succinctly  $p(x|\theta) = \theta^x(1 - \theta)^{1-x}$

- It's sensible to assume that  $\{x_1, \dots, x_N\}$  are **independent and identically distributed (i.i.d.)** Bernoullis.
- Thus the joint probability of the outcome  $\{x_1, \dots, x_N\}$  is

$$p(x_1, \dots, x_N|\theta) = \prod_{i=1}^N \theta^{x_i} (1 - \theta)^{1-x_i}$$

# Loss?

---

- We call the probability mass (or density for continuous) of the observed data the **likelihood function** (as a function of the parameters  $\theta$ ):

$$L(\theta) = \prod_{i=1}^N \theta^{x_i} (1 - \theta)^{1-x_i}$$

- We usually work with log-likelihoods:

$$\ell(\theta) = \sum_{i=1}^N x_i \log \theta + (1 - x_i) \log(1 - \theta)$$

- How can we choose  $\theta$ ? Good values of  $\theta$  should assign high probability to the observed data. This motivates the **maximum likelihood criterion**, that we should pick the parameters that maximize the likelihood:

$$\hat{\theta}_{\text{ML}} = \max_{\theta \in [0,1]} \ell(\theta)$$

# Maximum Likelihood Estimation for the Coin Example

---

- Remember how we found the optimal solution to linear regression by setting derivatives to zero? We can do that again for the coin example.

$$\begin{aligned}\frac{d\ell}{d\theta} &= \frac{d}{d\theta} \left( \sum_{i=1}^N x_i \log \theta + (1 - x_i) \log(1 - \theta) \right) \\ &= \frac{d}{d\theta} (N_H \log \theta + N_T \log(1 - \theta)) \\ &= \frac{N_H}{\theta} - \frac{N_T}{1 - \theta}\end{aligned}$$

where  $N_H = \sum_i x_i$  and  $N_T = N - \sum_i x_i$ .

- Setting this to zero gives the maximum likelihood estimate:

$$\hat{\theta}_{\text{ML}} = \frac{N_H}{N_H + N_T}.$$

# Maximum Likelihood Estimation

---

- Notice, in the coin example we are actually minimizing cross-entropies!

$$\begin{aligned}\hat{\theta}_{\text{ML}} &= \max_{\theta \in [0,1]} \ell(\theta) \\ &= \min_{\theta \in [0,1]} -\ell(\theta) \\ &= \min_{\theta \in [0,1]} \sum_{i=1}^N -x_i \log \theta - (1 - x_i) \log(1 - \theta)\end{aligned}$$

- This is an example of [maximum likelihood estimation](#).
  - define a model that assigns a probability (or has a probability density at) to a dataset
  - maximize the likelihood (or minimize the neg. log-likelihood).
- Many examples we've considered fall in this framework! Let's consider classification again.

# Discriminative VS Generative?

---

Two approaches to classification:

- **Discriminative approach**: estimate parameters of decision boundary/class separator directly from labeled examples.
  - Model  $p(t|\mathbf{x})$  directly (logistic regression models)
  - Learn mappings from inputs to classes (linear/logistic regression, decision trees etc)
  - Tries to solve: How do I separate the classes?
- **Generative approach**: model the distribution of inputs characteristic of the class (Bayes classifier).
  - Model  $p(\mathbf{x}|t)$
  - Apply Bayes Rule to derive  $p(t|\mathbf{x})$
  - Tries to solve: What does each class "look" like?
- Key difference: is there a distributional assumption over inputs?

# A Generative Model: Bayes Classifier

---

- Aim to classify text into spam/not-spam (yes  $c=1$ ; no  $c=0$ )
- Example: “You are one of the very few who have been selected as a winners for the free \$1000 Gift Card.”
- Use bag-of-words features, get binary vector  $\mathbf{x}$  for each email
- Vocabulary:
  - “a”: 1
  - ...
  - “car”: 0
  - “card”: 1
  - ...
  - “win”: 0
  - “winner”: 1
  - “winter”: 0
  - ...
  - “you”: 1



# Bayes Classifier

---

- Given features  $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$  we want to compute class probabilities using Bayes Rule:

$$\underbrace{p(c|\mathbf{x})}_{\text{Pr. class given words}} = \frac{p(\mathbf{x}, c)}{p(\mathbf{x})} = \frac{\overbrace{p(\mathbf{x}|c)}^{\text{Pr. words given class}} p(c)}{p(\mathbf{x})}$$

- More formally

$$\text{posterior} = \frac{\text{Class likelihood} \times \text{prior}}{\text{Evidence}}$$

- How can we compute  $p(\mathbf{x})$  for the two-class case? (Do we need to?)

$$p(\mathbf{x}) = p(\mathbf{x}|c=0)p(c=0) + p(\mathbf{x}|c=1)p(c=1)$$

- To compute  $p(c|\mathbf{x})$  we need:  $p(\mathbf{x}|c)$  and  $p(c)$

# Naive Bayes

---

- Assume we have two classes: spam and non-spam. We have a dictionary of  $D$  words, and binary features  $\mathbf{x} = [x_1, x_2, \dots, x_D]$  saying whether each word appears in the e-mail.
- If we define a joint distribution  $p(c, x_1, \dots, x_D)$ , this gives enough information to determine  $p(c)$  and  $p(\mathbf{x}|c)$ .
- Problem: specifying a joint distribution over  $D + 1$  binary variables requires  $2^{D+1} - 1$  entries. This is computationally prohibitive and would require an absurd amount of data to fit.
- We'd like to impose **structure** on the distribution such that:
  - it can be **compactly** represented
  - **learning** and **inference** are both tractable

# Naive Bayes

---

- Naive assumption: **Naive Bayes** assumes that the word features  $x_i$  are **conditionally independent** given the class  $c$ .
  - This means  $x_i$  and  $x_j$  are independent under the conditional distribution  $p(\mathbf{x}|c)$ .
  - Note: this doesn't mean they're independent.
  - Mathematically,

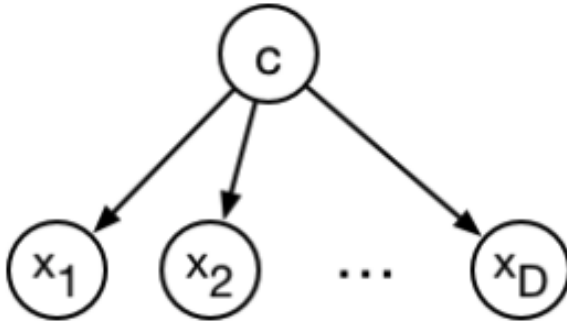
$$p(c, x_1, \dots, x_D) = p(c)p(x_1|c) \cdots p(x_D|c).$$

- Compact representation of the joint distribution
  - Prior probability of class:  $p(c = 1) = \pi$  (e.g. spam email)
  - Conditional probability of word feature given class:  $p(x_j = 1|c) = \theta_{jc}$  (e.g. word "price" appearing in spam)
  - $2D + 1$  parameters total (before  $2^{D+1} - 1$ )

# Bayes Nets

---

- We can represent this model using an [directed graphical model](#), or [Bayesian network](#):



- This graph structure means the joint distribution factorizes as a product of conditional distributions for each variable given its parent(s).
- Intuitively, you can think of the edges as reflecting a causal structure. But mathematically, this doesn't hold without additional assumptions.

# Naive Bayes: Learning

---

- The parameters can be learned efficiently because the log-likelihood decomposes into independent terms for each feature.

$$\begin{aligned}\ell(\boldsymbol{\theta}) &= \sum_{i=1}^N \log p(c^{(i)}, \mathbf{x}^{(i)}) = \sum_{i=1}^N \log \left\{ p(\mathbf{x}^{(i)} | c^{(i)}) p(c^{(i)}) \right\} \\ &= \sum_{i=1}^N \log \left\{ p(c^{(i)}) \prod_{j=1}^D p(x_j^{(i)} | c^{(i)}) \right\} \\ &= \sum_{i=1}^N \left[ \log p(c^{(i)}) + \sum_{j=1}^D \log p(x_j^{(i)} | c^{(i)}) \right] \\ &= \underbrace{\sum_{i=1}^N \log p(c^{(i)})}_{\text{Bernoulli log-likelihood of labels}} + \sum_{j=1}^D \underbrace{\sum_{i=1}^N \log p(x_j^{(i)} | c^{(i)})}_{\text{Bernoulli log-likelihood for feature } x_j}\end{aligned}$$

- Each of these log-likelihood terms depends on different sets of parameters, so they can be optimized independently.

# Naive Bayes: Learning

---

- We can handle these terms separately. For the prior we maximize:  $\sum_{i=1}^N \log p(c^{(i)})$
- This is a minor variant of our coin flip example. Let  $p(c^{(i)} = 1) = \pi$ . Note  $p(c^{(i)}) = \pi^{c^{(i)}}(1 - \pi)^{1-c^{(i)}}$
- Log-likelihood:

$$\sum_{i=1}^N \log p(c^{(i)}) = \sum_{i=1}^N c^{(i)} \log \pi + \sum_{i=1}^N (1 - c^{(i)}) \log(1 - \pi)$$

- Obtain MLEs by setting derivatives to zero:

$$\hat{\pi} = \frac{\sum_i \mathbb{I}[c^{(i)} = 1]}{N} = \frac{\# \text{ spams in dataset}}{\text{total } \# \text{ samples}}$$

# Naive Bayes: Learning

---

- Each  $\theta_{jc}$ 's can be treated separately: maximize  $\sum_{i=1}^N \log p(x_j^{(i)} | c^{(i)})$
- This is (again) a minor variant of our coin flip example. Let  $\theta_{jc} = p(x_j^{(i)} = 1 | c)$ . Note

$$p(x_j^{(i)} | c) = \theta_{jc}^{x_j^{(i)}} (1 - \theta_{jc})^{1-x_j^{(i)}}.$$

- Log-likelihood:

$$\begin{aligned} \sum_{i=1}^N \log p(x_j^{(i)} | c^{(i)}) &= \sum_{i=1}^N c^{(i)} \left\{ x_j^{(i)} \log \theta_{j1} + (1 - x_j^{(i)}) \log(1 - \theta_{j1}) \right\} \\ &\quad + \sum_{i=1}^N (1 - c^{(i)}) \left\{ x_j^{(i)} \log \theta_{j0} + (1 - x_j^{(i)}) \log(1 - \theta_{j0}) \right\} \end{aligned}$$

- Obtain MLEs by setting derivatives to zero:

$$\hat{\theta}_{jc} = \frac{\sum_i \mathbb{I}[x_j^{(i)} = 1 \ \& \ c^{(i)} = c]}{\sum_i \mathbb{I}[c^{(i)} = c]} \quad \text{for } \underline{c=1} \quad \frac{\text{\#word } j \text{ appears in spams}}{\text{\# spams in dataset}}$$

# Naive Bayes: Inference

---

- We predict the category by performing **inference** in the model.
- Apply **Bayes' Rule**:

$$p(c | \mathbf{x}) = \frac{p(c)p(\mathbf{x} | c)}{\sum_{c'} p(c')p(\mathbf{x} | c')} = \frac{p(c) \prod_{j=1}^D p(x_j | c)}{\sum_{c'} p(c') \prod_{j=1}^D p(x_j | c')}$$

- We need not compute the denominator if we're simply trying to determine the most likely  $c$ .
- Shorthand notation:

$$p(c | \mathbf{x}) \propto p(c) \prod_{j=1}^D p(x_j | c)$$

- For input  $\mathbf{x}$ , predict by comparing the values of  $p(c) \prod_{j=1}^D p(x_j | c)$  for different  $c$  (e.g. choose the largest).



# Naive Bayes

---

- Naive Bayes is an amazingly cheap learning algorithm!
- **Training time**: estimate parameters using maximum likelihood
  - Compute co-occurrence counts of each feature with the labels.
  - Requires only one pass through the data!
- **Test time**: apply Bayes' Rule
  - Cheap because of the model structure. (For more general models, Bayesian inference can be very expensive and/or complicated.)
- We covered the Bernoulli case for simplicity. But our analysis easily extends to other probability distributions.
- Unfortunately, it's usually less accurate in practice compared to discriminative models due to its "naive" independence assumption.

# MLE issue: Data Sparsity

---

- Maximum likelihood has a pitfall: if you have too little data, it can overfit.
- E.g., what if you flip the coin twice and get H both times?

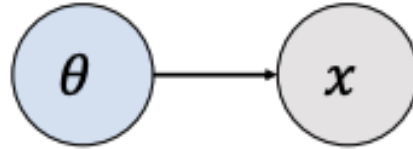
$$\theta_{\text{ML}} = \frac{N_H}{N_H + N_T} = \frac{2}{2 + 0} = 1$$

- Because it never observed T, it assigns this outcome probability 0. This problem is known as [data sparsity](#).

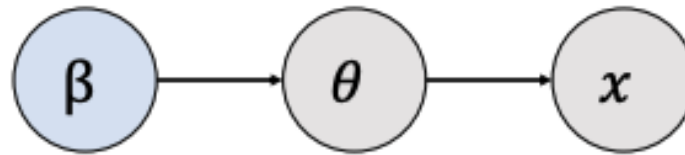
# Bayesian Parameter Estimation

---

- In maximum likelihood, the observations are treated as random variables, but the parameters are not.



- The **Bayesian** approach treats the parameters as random variables as well.  $\beta$  is the set of parameters in the prior distribution of  $\theta$ .



- To define a Bayesian model, we need to specify two distributions:
  - The **prior distribution**  $p(\theta)$ , which encodes our beliefs about the parameters *before* we observe the data
  - The **likelihood**  $p(\mathcal{D}|\theta)$ , same as in maximum likelihood

# Bayesian Parameter Estimation

---

- When we **update** our beliefs based on the observations, we compute the **posterior distribution** using Bayes' Rule:

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\boldsymbol{\theta})p(\mathcal{D} \mid \boldsymbol{\theta})}{\int p(\boldsymbol{\theta}')p(\mathcal{D} \mid \boldsymbol{\theta}') d\boldsymbol{\theta}'}.$$

- We rarely ever compute the denominator explicitly. In general, it is computationally intractable.

# Bayesian Parameter Estimation

---

- Let's revisit the coin example. We already know the likelihood:

$$L(\theta) = p(\mathcal{D}|\theta) = \theta^{N_H} (1 - \theta)^{N_T}$$

- It remains to specify the prior  $p(\theta)$ .
  - We can choose an **uninformative prior**, which assumes as little as possible. A reasonable choice is the uniform prior.
  - But our experience tells us 0.5 is more likely than 0.99. One particularly useful prior that lets us specify this is the **beta distribution**:

$$p(\theta; a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{a-1} (1-\theta)^{b-1}.$$

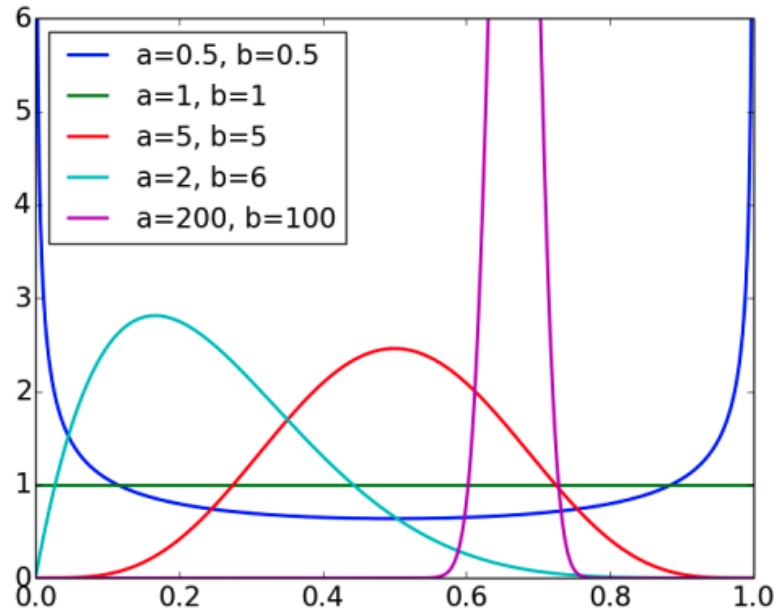
- This notation for proportionality lets us ignore the normalization constant:

$$p(\theta; a, b) \propto \theta^{a-1} (1-\theta)^{b-1}.$$

# Bayesian Parameter Estimation

---

- Beta distribution for various values of  $a, b$ :



- Some observations:
  - The expectation  $\mathbb{E}[\theta] = a/(a + b)$  (easy to derive).
  - The distribution gets more peaked when  $a$  and  $b$  are large.
  - The uniform distribution is the special case where  $a = b = 1$ .
- The beta distribution is used for is as a prior for the Bernoulli distribution.

# Bayesian Parameter Estimation

---

- Computing the posterior distribution:

$$\begin{aligned} p(\boldsymbol{\theta} \mid \mathcal{D}) &\propto p(\boldsymbol{\theta})p(\mathcal{D} \mid \boldsymbol{\theta}) \\ &\propto \left[ \theta^{a-1} (1 - \theta)^{b-1} \right] \left[ \theta^{N_H} (1 - \theta)^{N_T} \right] \\ &= \theta^{a-1+N_H} (1 - \theta)^{b-1+N_T}. \end{aligned}$$

- This is just a beta distribution with parameters  $N_H + a$  and  $N_T + b$ .
- The posterior expectation of  $\theta$  is:

$$\mathbb{E}[\theta \mid \mathcal{D}] = \frac{N_H + a}{N_H + N_T + a + b}$$

- The parameters  $a$  and  $b$  of the prior can be thought of as **pseudo-counts**.
  - The reason this works is that the prior and likelihood have the same functional form. This phenomenon is known as **conjugacy** (conjugate priors), and it's very useful.

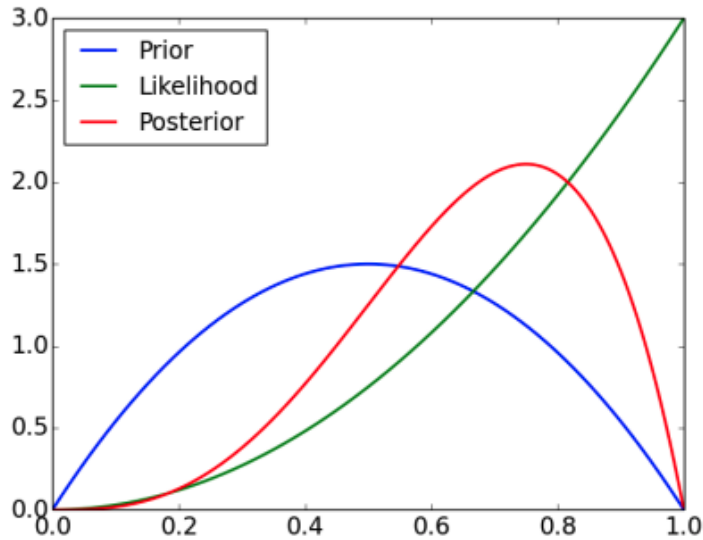
# Bayesian Parameter Estimation

---

Bayesian inference for the coin flip example:

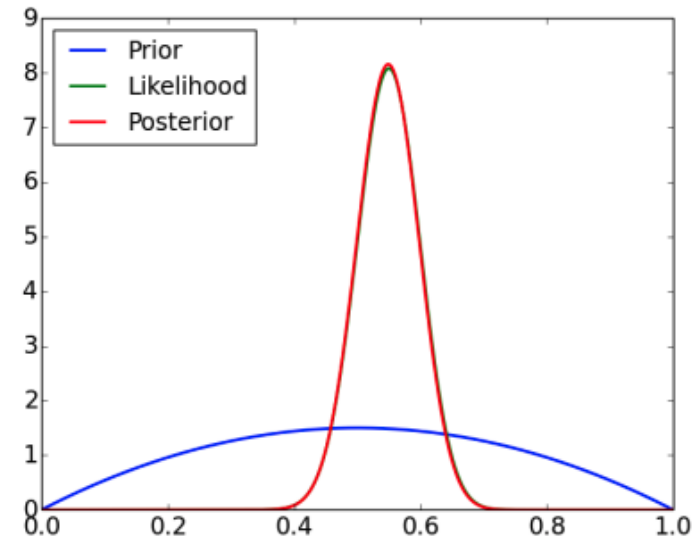
Small data setting

$$N_H = 2, N_T = 0$$



Large data setting

$$N_H = 55, N_T = 45$$



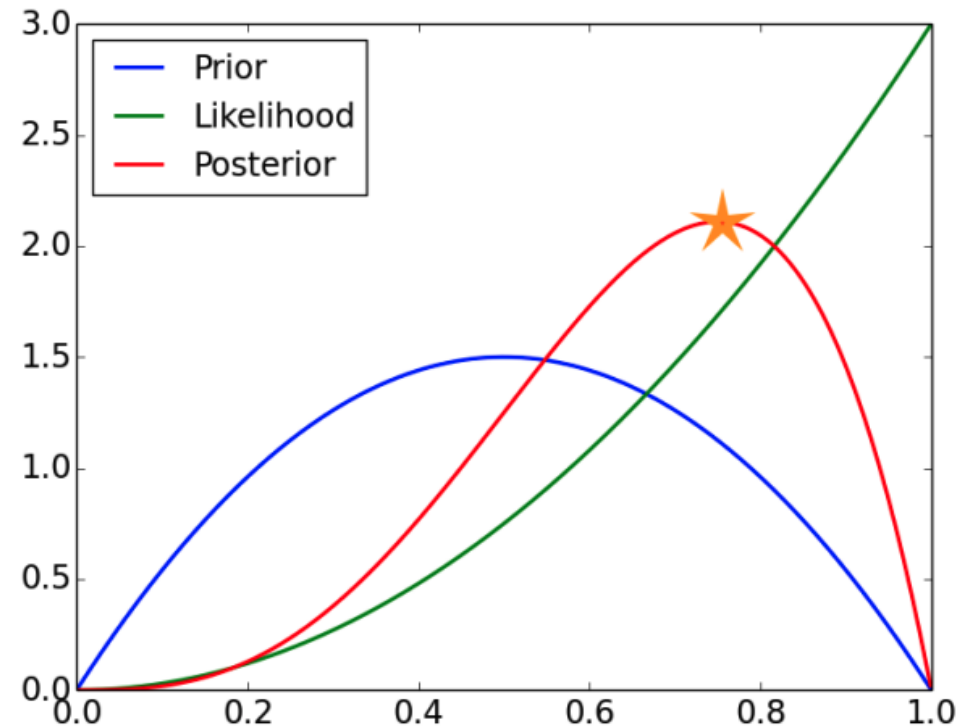
When you have enough observations, the **data overwhelm the prior**.



# Maximum A-Posteriori Estimation

---

Maximum a-posteriori (MAP) estimation: find the most likely parameter settings under the posterior



# Maximum A-Posteriori Estimation

---

- This converts the Bayesian parameter estimation problem into a maximization problem

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{\text{MAP}} &= \arg \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) \\ &= \arg \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}, \mathcal{D}) \\ &= \arg \max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}) p(\mathcal{D} \mid \boldsymbol{\theta}) \\ &= \arg \max_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}) + \log p(\mathcal{D} \mid \boldsymbol{\theta})\end{aligned}$$

# Maximum A-Posteriori Estimation

---

- Joint probability in the coin flip example:

$$\begin{aligned}\log p(\theta, \mathcal{D}) &= \log p(\theta) + \log p(\mathcal{D} | \theta) \\ &= \text{Const} + (a - 1) \log \theta + (b - 1) \log(1 - \theta) + N_H \log \theta + N_T \log(1 - \theta) \\ &= \text{Const} + (N_H + a - 1) \log \theta + (N_T + b - 1) \log(1 - \theta)\end{aligned}$$

- Maximize by finding a critical point

$$0 = \frac{d}{d\theta} \log p(\theta, \mathcal{D}) = \frac{N_H + a - 1}{\theta} - \frac{N_T + b - 1}{1 - \theta}$$

- Solving for  $\theta$ ,

$$\hat{\theta}_{\text{MAP}} = \frac{N_H + a - 1}{N_H + N_T + a + b - 2}$$

# Maximum A-Posteriori Estimation

---

Comparison of estimates in the coin flip example:

	Formula	$N_H = 2, N_T = 0$	$N_H = 55, N_T = 45$
$\hat{\theta}_{\text{ML}}$	$\frac{N_H}{N_H + N_T}$	1	$\frac{55}{100} = 0.55$
$\mathbb{E}[\theta \mathcal{D}]$	$\frac{N_H + a}{N_H + N_T + a + b}$	$\frac{4}{6} \approx 0.67$	$\frac{57}{104} \approx 0.548$
$\hat{\theta}_{\text{MAP}}$	$\frac{N_H + a - 1}{N_H + N_T + a + b - 2}$	$\frac{3}{4} = 0.75$	$\frac{56}{102} \approx 0.549$

$\hat{\theta}_{\text{MAP}}$  assigns nonzero probabilities as long as  $a, b > 1$ .