Precept 2: Classification

COS 484

Austin Wang

2/8 2023

Review Question

You train an n-gram model on some training corpus D using counts $P(w_n \mid w_1, \ldots, w_{n-1}) = \frac{c(w_1, \ldots, w_n)}{\sum_{v \in V} c(w_1, \ldots, w_{n-1}, v)}$. To prevent (possible) infinite perplexity on the test corpus D_t, you apply Laplace smoothing. Let $P(D), P(D_t)$ be the unsmoothed probabilities and $P'(D), P'(D_t)$ be the smoothed probabilities.

Review Question

You train an n-gram model on some training corpus D using counts $P(w_n \mid w_1, \ldots, w_{n-1}) = \dfrac{c(w_1, \ldots, w_n)}{c(w_1, \ldots, w_{n-1})}$. To prevent (possible) infinite perplexity on the test corpus D_t, you apply Laplace smoothing. Let $ppl(D), ppl(D_t)$ be perplexities of the unsmoothed model and $ppl'(D), ppl'(D_t)$ of the smoothed model. Is the following T, F or undetermined (depends on model, data, n, etc)?

1. $ppl'(D) \geq ppl(D)$

2. $ppl'(D_t) < ppl(D_t)$

3. $ppl(D) < ppl(D_t)$

Review Question

You train an n-gram model on some training corpus D using counts
$P(w_n \mid w_1, \ldots, w_{n-1}) = \dfrac{c(w_1, \ldots, w_n)}{c(w_1, \ldots, w_{n-1})}$. To prevent (possible) infinite perplexity on the test corpus D_t, you apply Laplace smoothing. Let $ppl(D), ppl(D_t)$ be perplexities of the unsmoothed model and $ppl'(D), ppl'(D_t)$ of the smoothed model. Is the following T, F or undetermined (depends on model, data, n, etc)?

1. $ppl'(D) \geq ppl(D)$
   This is true! Remember that setting the probability using counts (above) is the MLE estimate, which means that $P(D)$ cannot increase under any other distribution for $P(w_n \mid w_1, \ldots, w_{n-1})$

2. $ppl'(D_t) < ppl(D_t)$

3. $ppl(D) < ppl(D_t)$

Review Question

You train an n-gram model on some training corpus D using counts $P(w_n \mid w_1, \ldots, w_{n-1}) = \dfrac{c(w_1, \ldots, w_n)}{c(w_1, \ldots, w_{n-1})}$. To prevent (possible) infinite perplexity on the test corpus D_t, you apply Laplace smoothing. Let $ppl(D), ppl(D_t)$ be perplexities of the unsmoothed model and $ppl'(D), ppl'(D_t)$ of the smoothed model. Is the following T, F or undetermined (depends on model, data, n, etc)?

1. $ppl'(D) \geq ppl(D)$ - T

2. $ppl'(D_t) < ppl(D_t)$

   This is undetermined! It's not clear that the test corpus will have infinite perplexity. It is possible that the test corpus is very similar to the train corpus, and smoothing will cause its probability to drop.

3. $ppl(D) < ppl(D_t)$

Review Question

You train an n-gram model on some training corpus D using counts $P(w_n \mid w_1, \ldots, w_{n-1}) = \dfrac{c(w_1, \ldots, w_n)}{c(w_1, \ldots, w_{n-1})}$. To prevent (possible) infinite perplexity on the test corpus D_t, you apply Laplace smoothing. Let $ppl(D), ppl(D_t)$ be perplexities of the unsmoothed model and $ppl'(D), ppl'(D_t)$ of the smoothed model. Is the following T, F or undetermined (depends on model, data, n, etc)?

1. $ppl'(D) \geq ppl(D)$ - T

2. $ppl'(D_t) < ppl(D_t)$ - U

3. $ppl(D) < ppl(D_t)$
   Undetermined. A test corpus consisting solely of high-frequency n-grams might have a higher probability

Todays Topics

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$. Two ways to do this:

Naive Bayes

Logistic Regression

Naive Bayes Intuition

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$ (the MAP estimate)

Naive Bayes Intuition

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$ (the MAP estimate)

Language Model: $P(w_1, \ldots, w_K) = P(d)$ gives us a probability for a text sequence

Conditional Language Model: $P(d \mid c)$ gives us probability of a text sequence conditioned on something

Naive Bayes Intuition

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$ (the MAP estimate)

Language Model: $P(w_1, \ldots, w_K) = P(d)$ gives us a probability for a text sequence

Conditional Language Model: $P(d \mid c)$ gives us probability of a text sequence conditioned on something

Given a conditional language model (which conditions on the classes) we can find the MAP estimate using Bayes rule!

Naive Bayes Intuition

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$ (the MAP estimate)

Language Model: $P(w_1, \ldots, w_K) = P(d)$ gives us a probability for a text sequence

Conditional Language Model: $P(d \mid c)$ gives us probability of a text sequence conditioned on something

Given a conditional language model (which conditions on the classes) we can find the MAP estimate using Bayes rule!

$$c_{\text{MAP}} = \text{argmax}_{c \in C} P(c \mid d)$$

$$= \text{argmax}_{c \in C} \frac{P(d \mid c) P(c)}{P(d)}$$

$$= \text{argmax}_{c \in C} P(d \mid c) P(c)$$

Naive Bayes: An "illustration"

Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

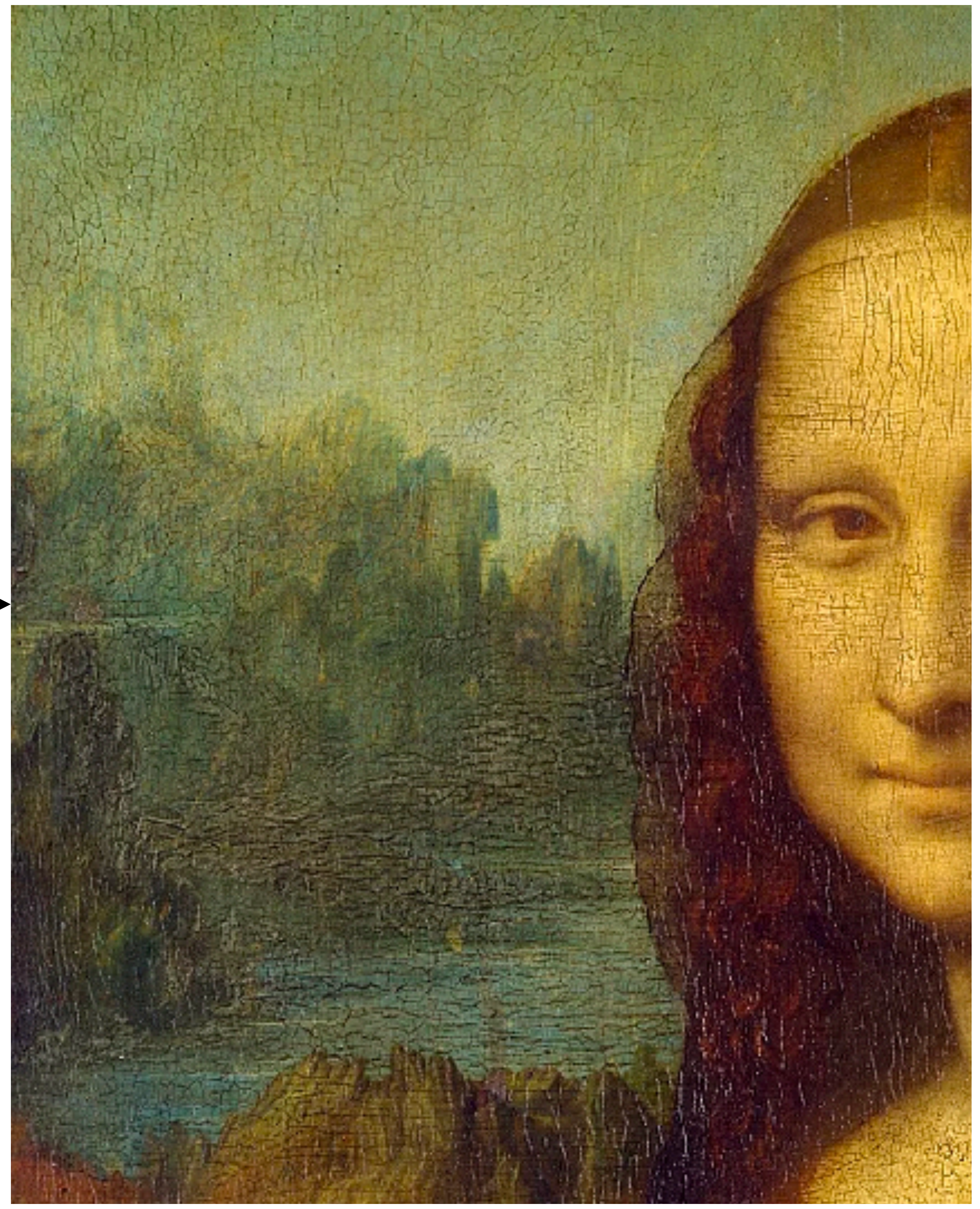
Let's say you work on a group project with a friend, and we want a model that can attribute your writing vs your friend's writing. C = {you, friend}

Naive Bayes: An "illustration"

Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

Let's say you work on a group project with a friend, and we want a model that can attribute your writing vs your friend's writing. C = {you, friend}

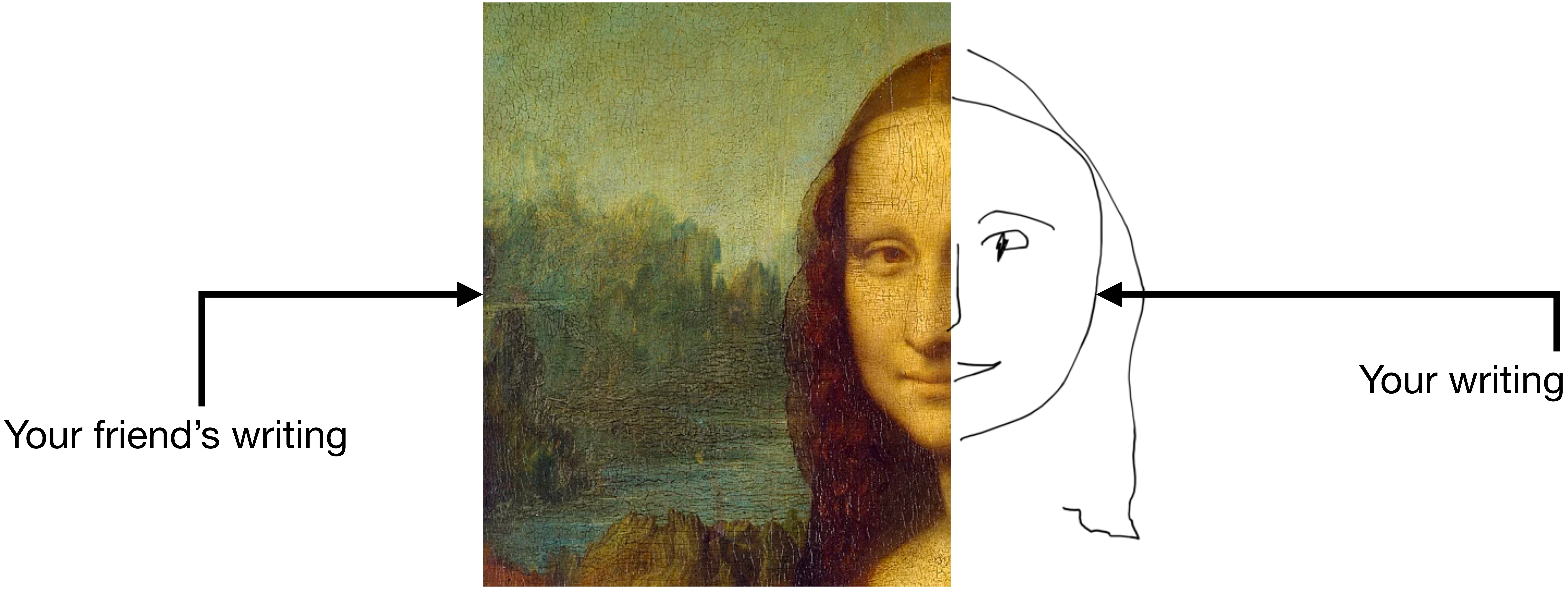
Your friend's writing



Naive Bayes: An "illustration"

Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

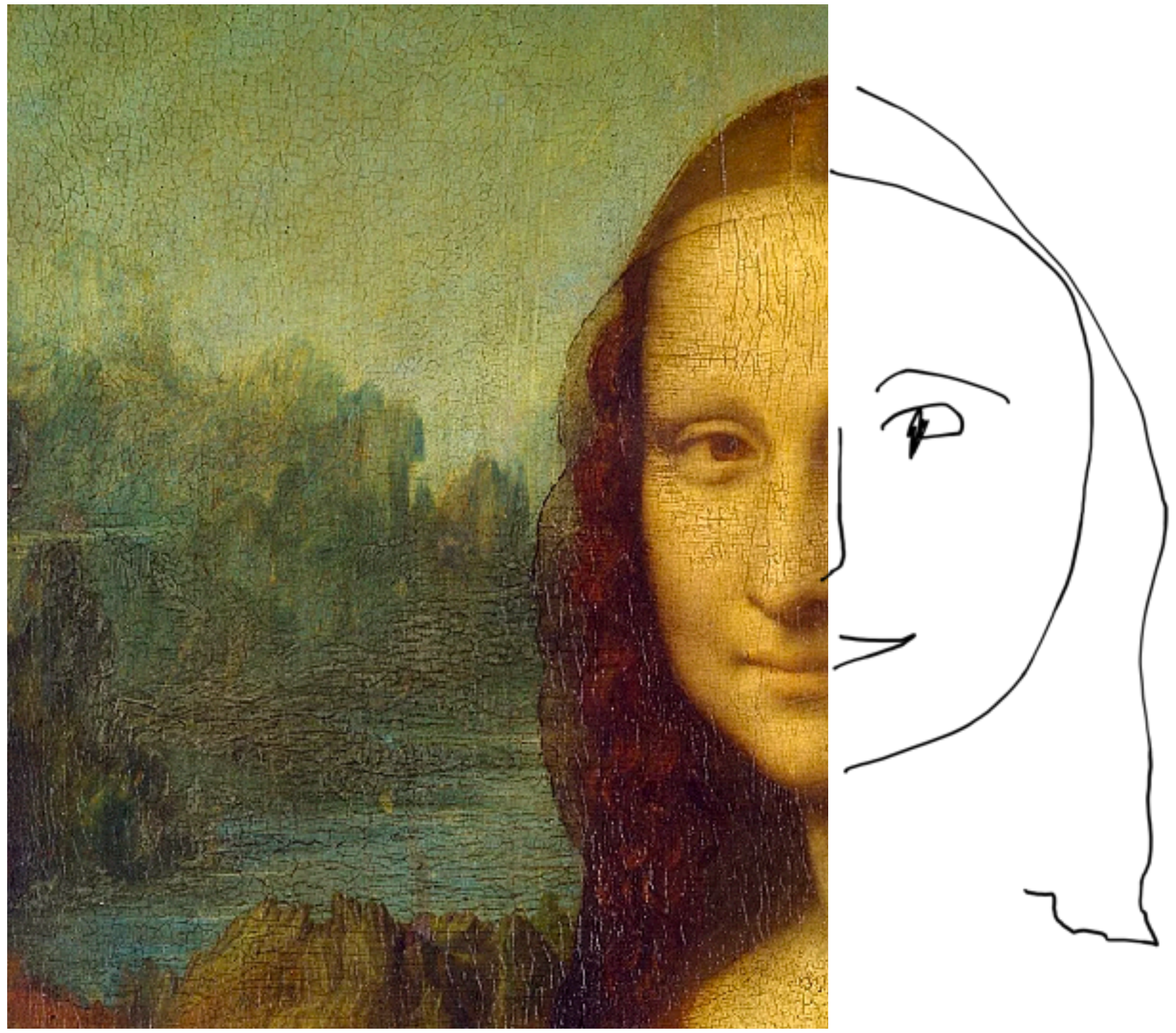
Let's say you work on a group project with a friend, and we want a model that can attribute your writing vs your friend's writing. C = {you, friend}



Naive Bayes: An "illustration"

Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c) \boxed{P(c)}$

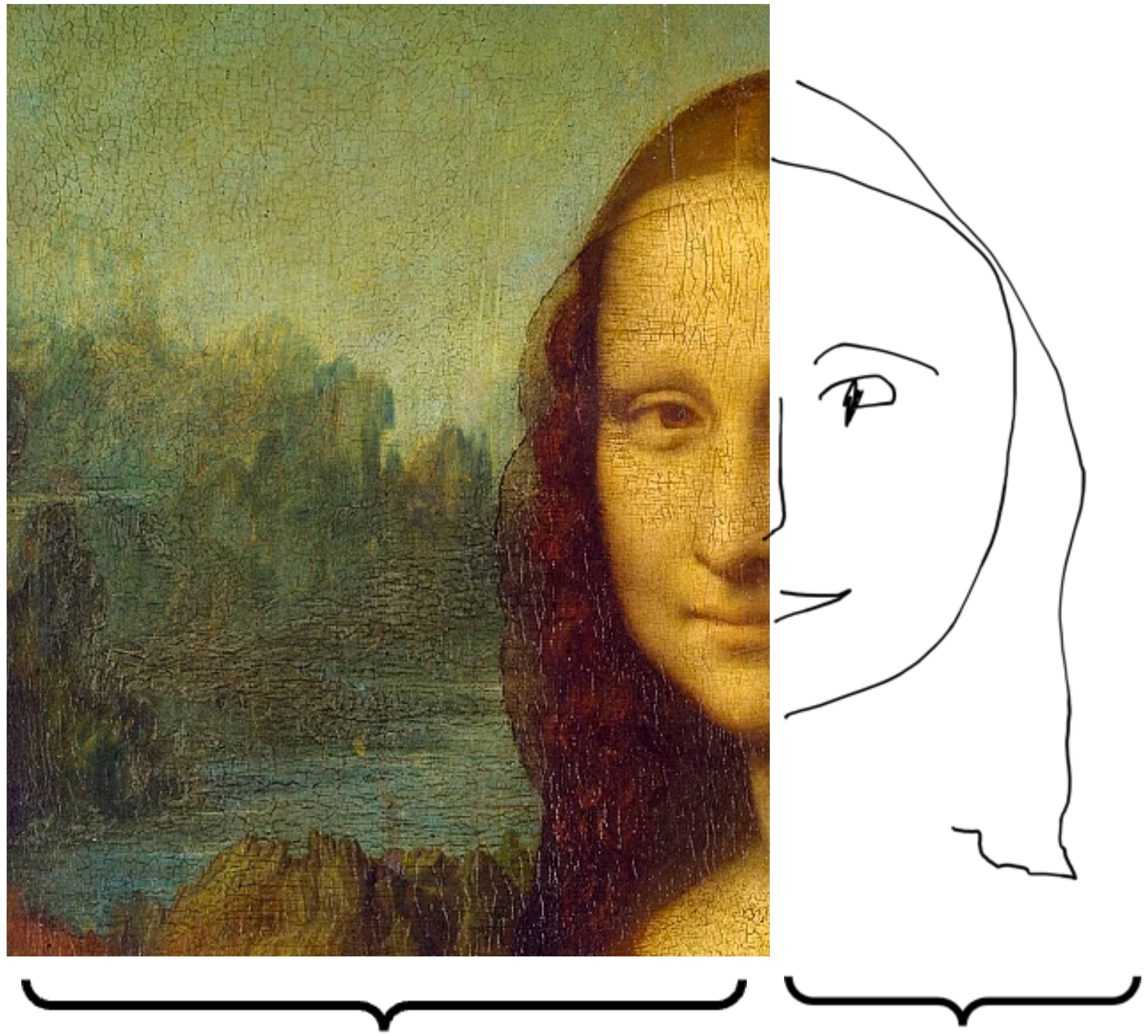
First, we determine who did more work. This gives us a prior estimate (bias) on whether any document was written by you or your friend.



Naive Bayes: An "illustration"

Summary: We want to find the class $c_{MAP} = \underset{c \in C}{\arg\max}\, P(d \mid c) \boxed{P(c)}$

First, we determine who did more work. This gives us a prior estimate (bias) on whether any document was written by you or your friend.

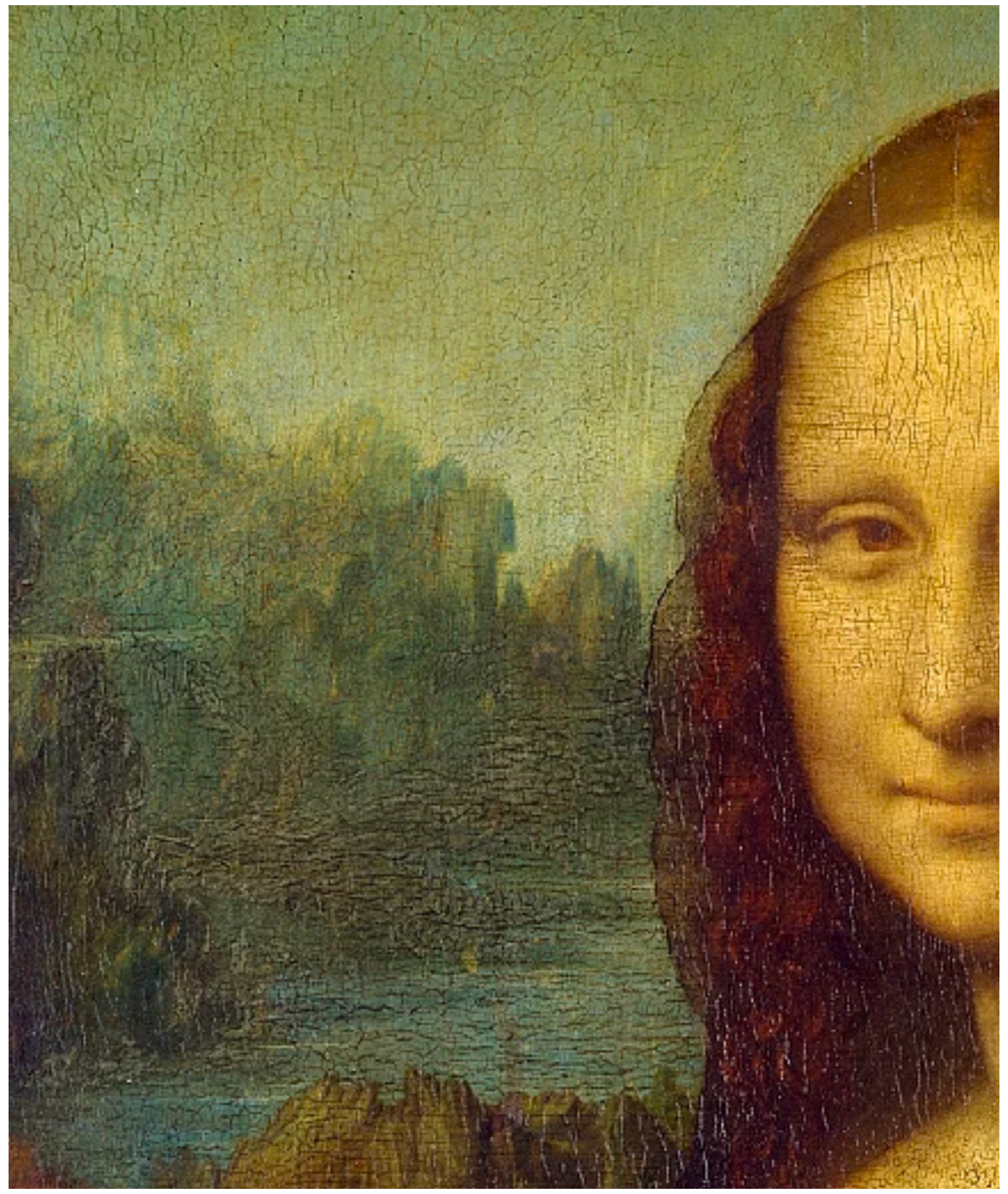


P(friend) = 3/4 P(you) = 1/4

Naive Bayes: An "illustration"

Summary: We want to find the class $c_{MAP} = \underset{c \in C}{\arg\max}\, \boxed{P(d \mid c)} P(c)$

Now, to compute $P(d \mid c)$ for any input document d We can train two language models, one trained on your writing, and one on your friend's



P(x|friend)

P(x|you)

Naive Bayes: An "illustration"

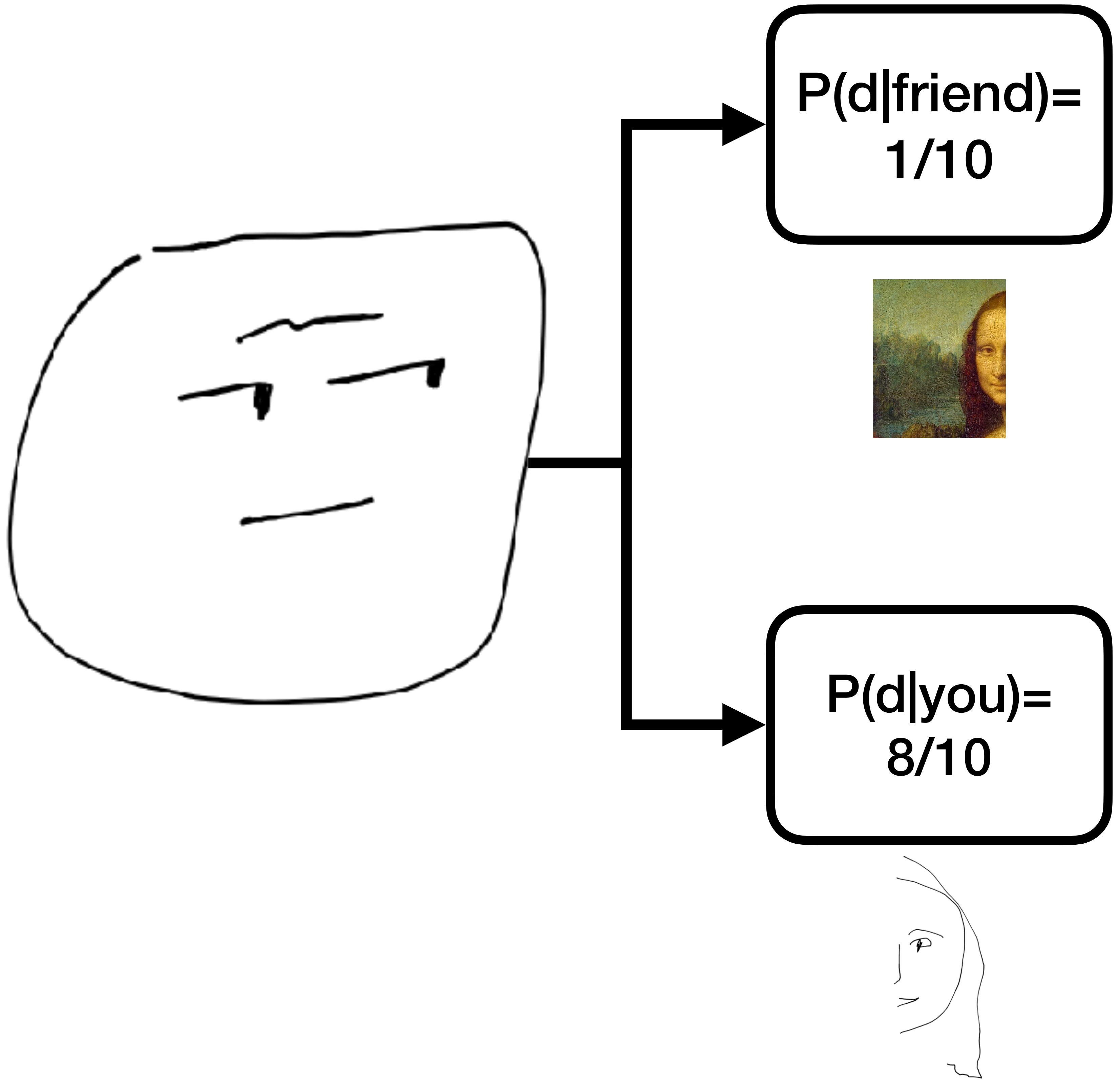
Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

Now given a new sample d, we can compute the probability under each LM to find $P(d \mid c)$. And multiply this by $P(c)$ to find the MAP estimate.

Naive Bayes: An "illustration"

Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

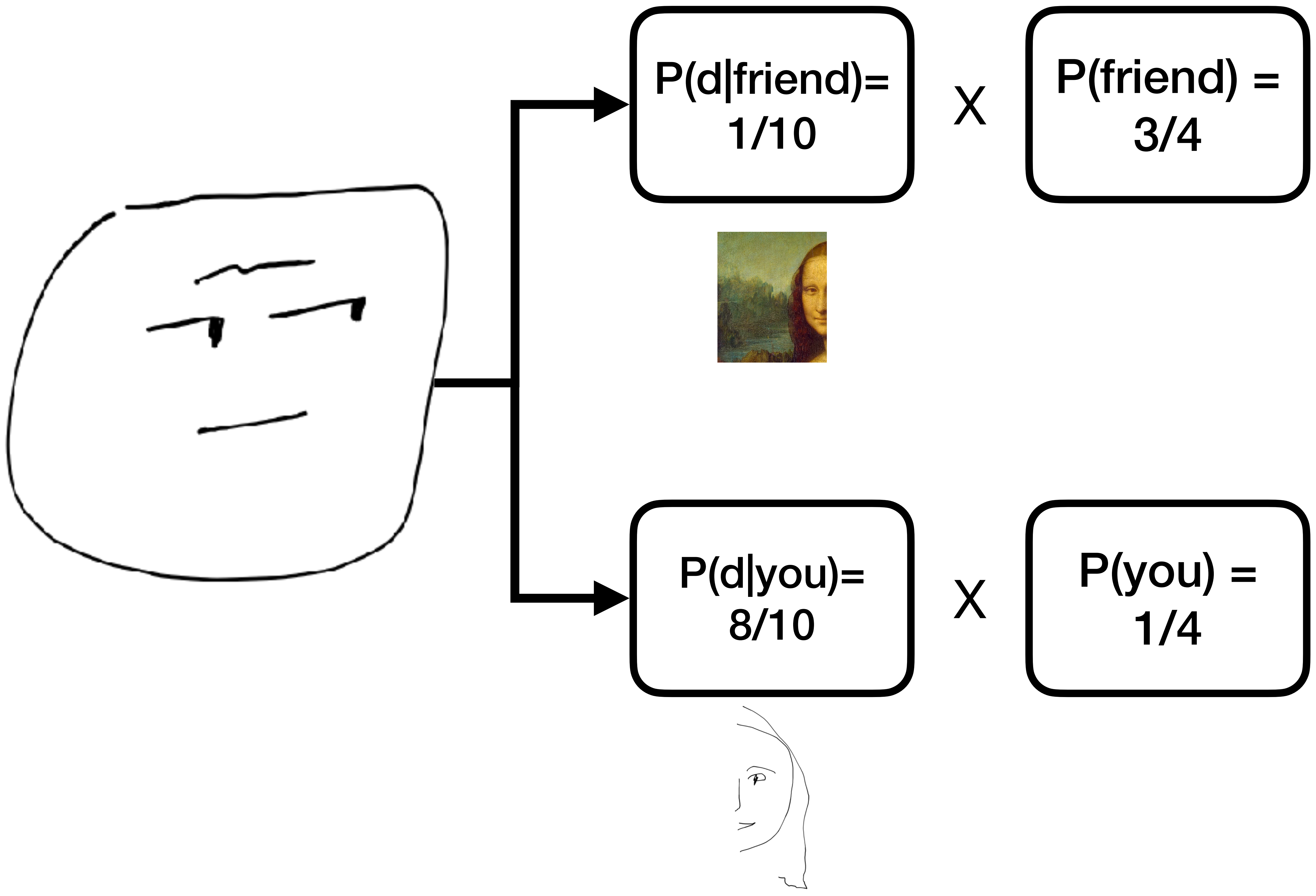
Now given a new sample d, we can compute the probability under each LM to find $P(d \mid c)$. And multiply this by $P(c)$ to find the MAP estimate.



Naive Bayes: An "illustration"

Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

Now given a new sample d, we can compute the probability under each LM to find $P(d \mid c)$. And multiply this by $P(c)$ to find the MAP estimate.

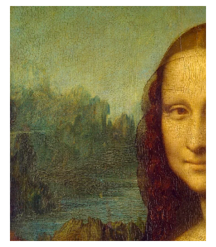


Naive Bayes: An “illustration”

Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

Now given a new sample d, we can compute the probability under each LM to find $P(d \mid c)$. And multiply this by $P(c)$ to find the MAP estimate.

P(d|friend)= 1/10 X P(friend) = 3/4 = 3 / 40



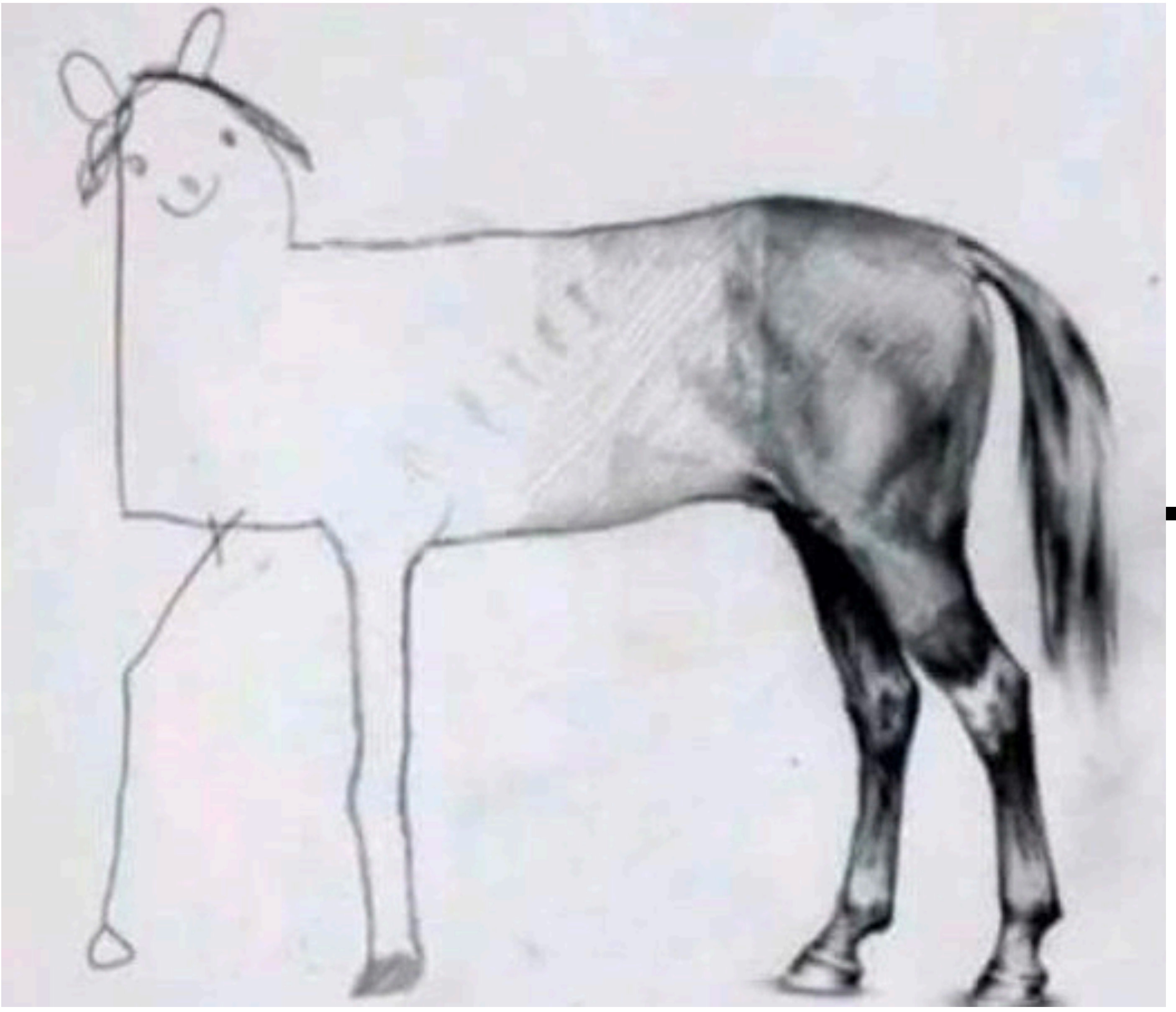
P(d|you)= 8/10 X P(you) = 1/4 = 8 / 40

MAP estimate is you!

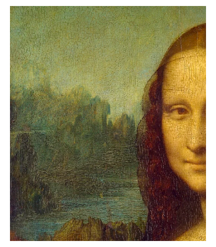
Naive Bayes: An "illustration"

Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

The prior is important when the probabilities are close under each LM!



P(d|friend)= 1/10 X P(friend) = 3/4 = 3 / 40



MAP estimate is friend!

P(d|you)= 1/10 X P(you) = 1/4 = 1 / 40

Naive Bayes: One extra detail…

Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c) P(c)$

Now, to compute $P(d \mid c)$ for any input document d We can train two language models, one trained on your writing, and one on your friend's

P(x|you)

$$P(w_1, w_2, \ldots, w_K \mid c) = P(w_1 \mid c) P(w_2 \mid c) \ldots P(w_K \mid c)$$

Naive Bayes: One extra detail…

Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d|c)P(c)$

Now, to compute $P(d|c)$ for any input document d We can train two language models, one trained on your writing, and one on your friend's

P(x|you)

$$P(w_1, w_2, \ldots, w_K | c) = P(w_1|c)P(w_2|c)\ldots P(w_K|c)$$

To simplify our LM, we use unigrams. This is equivalent to saying, we assume all words are **independent** of each other. This is the "naive" assumption of Naive Bayes

Naive Bayes: Summary

Naive Bayes: Summary

1. Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c that maximizes $P(c \mid d)$.

Naive Bayes: Summary

1. Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c that maximizes $P(c \mid d)$.

2. We don't know $P(c \mid d)$, but we know how to estimate $P(d \mid c)$ using a simple LM! → we can get $P(c \mid d)$ using Bayes' rule!
    1. $P(c \mid d) \propto P(d \mid c)P(c)$

Naive Bayes: Summary

1. Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c that maximizes $P(c \mid d)$.
2. We don't know $P(c \mid d)$, but we know how to estimate $P(d \mid c)$ using a simple LM! $\rightarrow$ we can get $P(c \mid d)$ using Bayes' rule!
    1. $P(c \mid d) \propto P(d \mid c)P(c)$
3. Bayes rule requires us to estimate $P(c)$, we can do this just by counting the proportion of documents that are class c

Naive Bayes: Summary

1. Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c that maximizes $P(c \mid d)$.
2. We don't know $P(c \mid d)$, but we know how to estimate $P(d \mid c)$ using a simple LM! $\rightarrow$ we can get $P(c \mid d)$ using Bayes' rule!
   1. $P(c \mid d) \propto P(d \mid c)P(c)$
3. Bayes rule requires us to estimate $P(c)$, we can do this just by counting the proportion of documents that are class c
4. To estimate $P(d \mid c)$ let's be lazy and choose the simplest possible LM that assume (Naively) that each word is independent - the unigram

Naive Bayes: Summary

1. Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c that maximizes $P(c \mid d)$.
2. We don't know $P(c \mid d)$, but we know how to estimate $P(d \mid c)$ using a simple LM! $\rightarrow$ we can get $P(c \mid d)$ using Bayes' rule!
    1. $P(c \mid d) \propto P(d \mid c)P(c)$
3. Bayes rule requires us to estimate $P(c)$, we can do this just by counting the proportion of documents that are class c
4. To estimate $P(d \mid c)$ let's be lazy and choose the simplest possible LM that assume (Naively) that each word is independent - the unigram
5. Combine 3 + 4 and you can find the MAP estimate: $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

Advantages of Naive Bayes

- Very fast, low storage requirements
- Robust to irrelevant features

  Irrelevant features cancel each other without affecting results

- Optimal if the independence assumptions hold

  If assumed independence is correct, this is the 'Bayes optimal' classifier

- A good dependable baseline for text classification

  However, other classifiers can give better accuracy

Todays Topics

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$. Two ways to do this:

Naive Bayes

Logistic Regression

Logistic Regression: Intuition

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$

Logistic Regression: Intuition

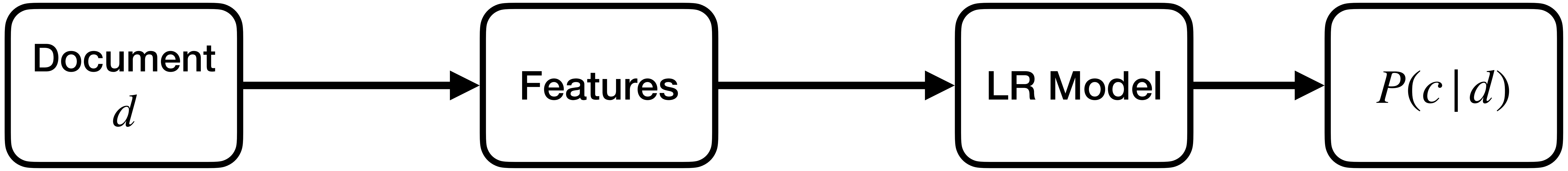
Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$

Compared to NB, with LR we take a more direct approach: directly compute $P(c \mid d)$ given a set of features constructed from the input document d.

Logistic Regression: Intuition

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$

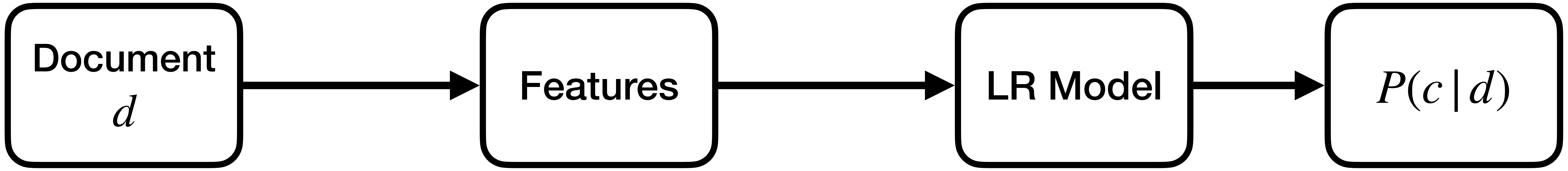
Compared to NB, with LR we take a more direct approach: directly compute $P(c \mid d)$ given a set of features constructed from the input document d.



Logistic Regression: Features

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$

Compared to NB, with LR we take a more direct approach: directly compute $P(c \mid d)$ given a set of features constructed from the input document d.



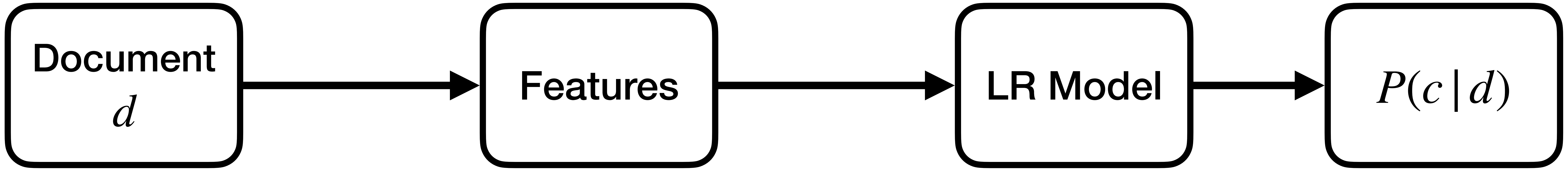
Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

This is the feature vector x for some input document d

Logistic Regression: Features

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$

Compared to NB, with LR we take a more direct approach: directly compute $P(c \mid d)$ given a set of features constructed from the input document d.



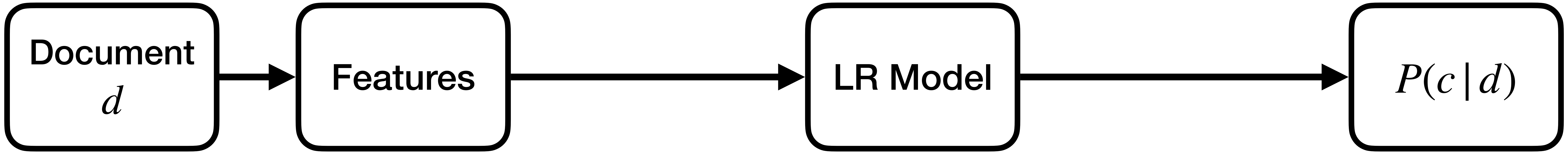
Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

The features to use is a design decision. A natural default is to use a vector $x \in \mathbb{R}^{|V|}$ where each dim is the counts of one word in the vocabulary. (BOW)

Logistic Regression: LR Model

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$

Now given some feature vector x how do we turn this to a probability?

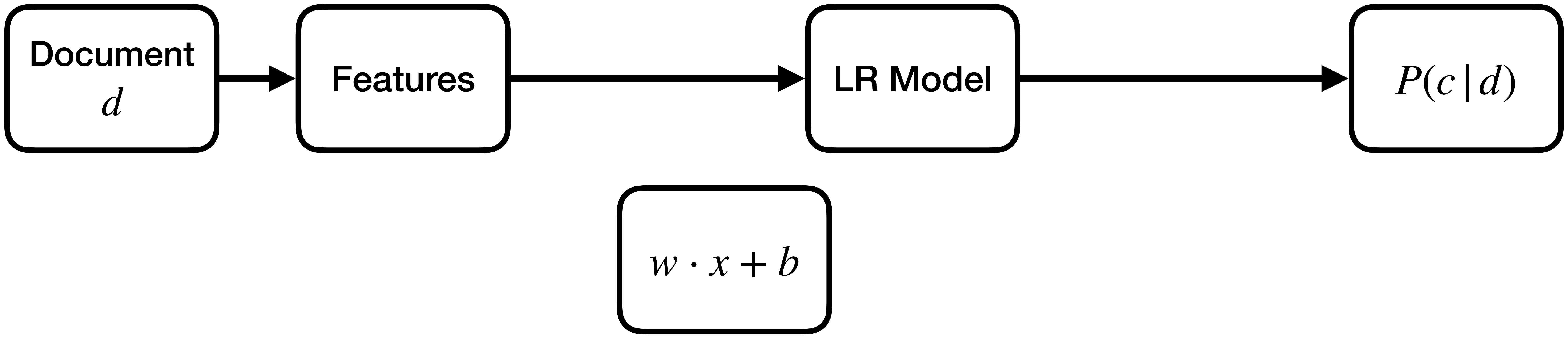


Logistic Regression: LR Model

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$

Now given some feature vector x how do we turn this to a probability?

1. Convert the features to a number. The higher the number, the more confident we are that the document belongs to a class. We call these numbers **logits.**

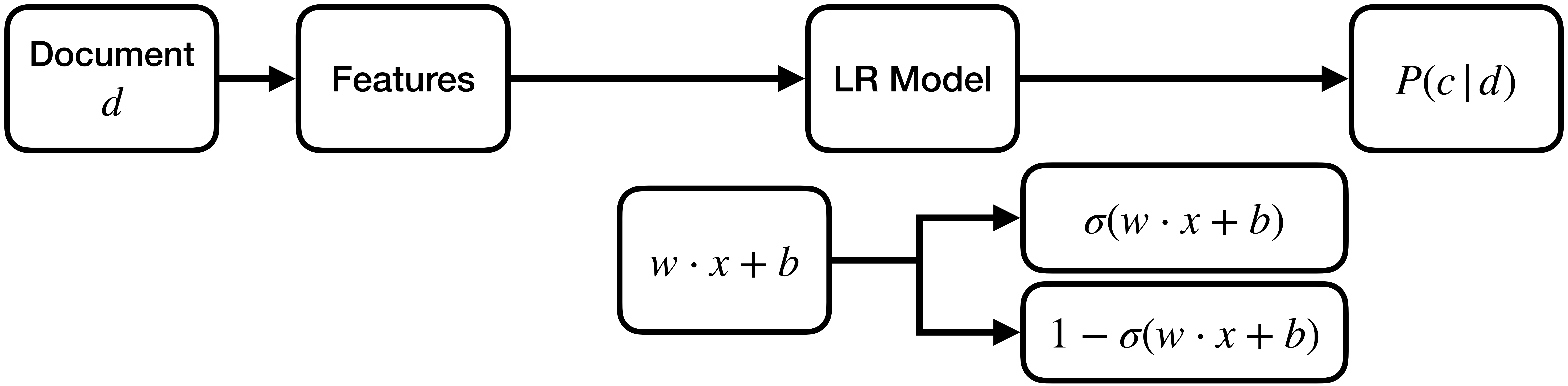


Logistic Regression: LR Model

Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$

Now given some feature vector x how do we turn this to a probability?

1. Convert the features to a number. The higher the number, the more confident we are that the document belongs to a class. We call these numbers **logits.**
2. Normalize the logits using sigmoid so we get a well-defined probability distribution.
    1. For more than 2 classes we use the softmax, which is the m > 2 generalization of sigmoid



Logistic Regression: How do we set w, b ?

Summary: we want to estimate $P(c \mid d)$ using a model $\sigma(w \cdot x + b)$

Logistic Regression: How do we set w, b ?

Summary: we want to estimate $P(c \mid d)$ using a model $\sigma(w \cdot x + b)$

We use GD. But what should we make the loss?

Logistic Regression: How do we set w, b ?

Summary: we want to estimate $P(c \mid d)$ using a model $\sigma(w \cdot x + b)$

We use GD. But what should we make the loss?

Let our train dataset be $\mathcal{D} = \{(d_1, c_1), \ldots, (d_n, c_n)\}$. Let's find the probability of seeing these documents and labels. Assume that each datapoint is independent of the other.

Logistic Regression: How do we set w, b ?

Summary: we want to estimate $P(c \mid d)$ using a model $\sigma(w \cdot x + b)$

We use GD. But what should we make the loss?

Let our train dataset be $\mathscr{D} = \{(d_1, c_1), \ldots, (d_n, c_n)\}$. Let's find the probability of seeing these documents and labels. Assume that each datapoint is independent of the other.

$$P(\mathscr{D}) = P(c_1 \mid d_1) \cdots P(c_n \mid d_n) = \Pi_i P(c_i \mid d_i)$$

Logistic Regression: How do we set w, b ?

Summary: we want to estimate $P(c \mid d)$ using a model $\sigma(w \cdot x + b)$

We use GD. But what should we make the loss?

Let our train dataset be $\mathscr{D} = \{(d_1, c_1), \ldots, (d_n, c_n)\}$. Let's find the probability of seeing these documents and labels. Assume that each datapoint is independent of the other.

$$P(\mathscr{D}) = P(c_1 \mid d_1) \cdots P(c_n \mid d_n) = \Pi_i P(c_i \mid d_i)$$

How to set $\theta = (w, b)$?

Logistic Regression: How do we set w, b ?

Summary: we want to estimate $P(c \mid d)$ using a model $\sigma(w \cdot x + b)$

We use GD. But what should we make the loss?

Let our train dataset be $\mathscr{D} = \{(d_1, c_1), \ldots, (d_n, c_n)\}$. Let's find the probability of seeing these documents and labels. Assume that each datapoint is independent of the other.

$$P(\mathscr{D}) = P(c_1 \mid d_1) \cdots P(c_n \mid d_n) = \Pi_i P(c_i \mid d_i)$$

How to set $\theta = (w, b)$? Use the MLE principle! Set θ such that $P(\mathscr{D})$ is maximized. This is analogous to setting the n-gram probabilities such that the probability of the train corpus is maximal.

Logistic Regression: How do we set w, b ?

Summary: we want to estimate $P(c \mid d)$ using a model $\sigma(w \cdot x + b)$

We use GD. But what should we make the loss?

Let our train dataset be $\mathscr{D} = \{(d_1, c_1), \ldots, (d_n, c_n)\}$. Let's find the probability of seeing these documents and labels. Assume that each datapoint is independent of the other.

$$P(\mathscr{D}) = P(c_1 \mid d_1) \cdots P(c_n \mid d_n) = \Pi_i P(c_i \mid d_i)$$

How to set $\theta = (w, b)$? Use the MLE principle! Set θ such that $P(\mathscr{D})$ is maximized. This is analogous to setting the n-gram probabilities such that the probability of the train corpus is maximal.

So we can directly use GD to minimize: $-\Pi_i P(c_i \mid d_i)$

Logistic Regression: How do we set w, b ?

Summary: we want to estimate $P(c \mid d)$ using a model $\sigma(w \cdot x + b)$

We use GD. But what should we make the loss?

Let our train dataset be $\mathscr{D} = \{(d_1, c_1), \ldots, (d_n, c_n)\}$. Let's find the probability of seeing these documents and labels. Assume that each datapoint is independent of the other.

$$P(\mathscr{D}) = P(c_1 \mid d_1) \cdots P(c_n \mid d_n) = \Pi_i P(c_i \mid d_i)$$

How to set $\theta = (w, b)$? Use the MLE principle! Set θ such that $P(\mathscr{D})$ is maximized. This is analogous to setting the n-gram probabilities such that the probability of the train corpus is maximal.

So we can directly use GD to minimize: $-\Pi_i P(c_i \mid d_i)$

Since $\log$ is monotonic, this is equivalent to minimizing: $-\sum_i \log P(c_i \mid d_i)$ $\leftarrow$ this is just CE loss!

Logistic Regression: How do we set w, b ?

Summary: we want to estimate $P(c \mid d)$ using a model $\sigma(w \cdot x + b)$

Want to minimize: $-\sum \log P(c_i \mid d_i)$ $\leftarrow$ this is just CE loss!

- Loss: $-\log \prod_{i=1}^{n} P(y_i \mid x_i) = -\sum_{i=1}^{n} \log P(y_i \mid x_i)$

$$L_{CE} = -\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

Logistic Regression: How do we set w, b ?

Summary: we want to estimate $P(c \mid d)$ using a model $\sigma(w \cdot x + b)$

Want to minimize: $-\sum \log P(c_i \mid d_i)$ ← this is just CE loss!

- Loss: $-\log \prod_{i=1}^{n} P(y_i \mid x_i) = -\sum_{i=1}^{n} \log P(y_i \mid x_i)$

$$L_{CE} = -\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$$

- Gradient, $\frac{dL_{CE}(\mathbf{w}, b)}{dw_j} = \sum_{i=1}^{n} [\hat{y}_i - y_i]x_{i,j}$

The j-th value of the feature vector $\mathbf{x}_i$

- $\frac{dL_{CE}(\mathbf{w}, b)}{db} = \sum_{i=1}^{n} [\hat{y}_i - y_i]$

Logistic Regression: Summary

Logistic Regression: Summary

1. Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c that maximizes $P(c \mid d)$. Let's say we estimating $P(d \mid c)$ reliably is hard, we will need to estimate $P(c \mid d)$ directly.

Logistic Regression: Summary

1. Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c that maximizes $P(c \mid d)$. Let's say we estimating $P(d \mid c)$ reliably is hard, we will need to estimate $P(c \mid d)$ directly.
2. Want to turn d into a vector x because then we can operate on it more conveniently.
    1. We can use a BOW, where each dim in $x \in \mathbb{R}^{|V|}$ is the # of times a word in V appears
    2. We can also be creative and add additional features we think are important (e.g. # of emojis in text)

Logistic Regression: Summary

1. Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c that maximizes $P(c \mid d)$. Let's say we estimating $P(d \mid c)$ reliably is hard, we will need to estimate $P(c \mid d)$ directly.
2. Want to turn d into a vector x because then we can operate on it more conveniently.
    1. We can use a BOW, where each dim in $x \in \mathbb{R}^{|V|}$ is the # of times a word in V appears
    2. We can also be creative and add additional features we think are important (e.g. # of emojis in text)
3. Somehow we need to turn x into a single number, because $P(c \mid d)$ is a single number.
    1. Let's be as lazy as possible and just take a linear combination of the features: $w \cdot x + b$

Logistic Regression: Summary

1. Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c that maximizes $P(c \mid d)$. Let's say we estimating $P(d \mid c)$ reliably is hard, we will need to estimate $P(c \mid d)$ directly.
2. Want to turn d into a vector x because then we can operate on it more conveniently.
    1. We can use a BOW, where each dim in $x \in \mathbb{R}^{|V|}$ is the # of times a word in V appears
    2. We can also be creative and add additional features we think are important (e.g. # of emojis in text)
3. Somehow we need to turn x into a single number, because $P(c \mid d)$ is a single number.
    1. Let's be as lazy as possible and just take a linear combination of the features: $w \cdot x + b$
4. Oh no! The linear combination might not be in $[0,1]$, so we normalize using sigmoid: $\sigma(x) = (1 + e^{-x})^{-1}$
    1. The probability for one class is $\sigma(w \cdot x + b)$, so the other class must have prob $1 - \sigma(w \cdot x + b)$

Logistic Regression: Summary

1. Given a document $d = w_1, ..., w_K$ and a set of classes $\mathscr{C} = \{c_1, ..., c_m\}$, we want to find the class c that maximizes $P(c \mid d)$. Let's say we estimating $P(d \mid c)$ reliably is hard, we will need to estimate $P(c \mid d)$ directly.
2. Want to turn d into a vector x because then we can operate on it more conveniently.
    1. We can use a BOW, where each dim in $x \in \mathbb{R}^{|V|}$ is the # of times a word in V appears
    2. We can also be creative and add additional features we think are important (e.g. # of emojis in text)
3. Somehow we need to turn x into a single number, because $P(c \mid d)$ is a single number.
    1. Let's be as lazy as possible and just take a linear combination of the features: $w \cdot x + b$
4. Oh no! The linear combination might not be in $[0,1]$, so we normalize using sigmoid: $\sigma(x) = (1 + e^{-x})^{-1}$
    1. The probability for one class is $\sigma(w \cdot x + b)$, so the other class must have prob $1 - \sigma(w \cdot x + b)$
5. Given our model, we can estimate the probability of a train set under the model $P(\mathscr{D})$
    1. We will set w, b so that $P(\mathscr{D})$ = $\Pi_i P(c_i \mid d_i)$ is maximal (MLE principle)
    2. For stability and convenience we can take the $\log$ to minimize $-\sum_i \log P(c_i \mid d_i)$ this is CE loss

Logistic Regression: Summary

1. Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c that maximizes $P(c \mid d)$. Let's say we estimating $P(d \mid c)$ reliably is hard, we will need to estimate $P(c \mid d)$ directly.
2. Want to turn d into a vector x because then we can operate on it more conveniently.
    1. We can use a BOW, where each dim in $x \in \mathbb{R}^{|V|}$ is the # of times a word in V appears
    2. We can also be creative and add additional features we think are important (e.g. # of emojis in text)
3. Somehow we need to turn x into a single number, because $P(c \mid d)$ is a single number.
    1. Let's be as lazy as possible and just take a linear combination of the features: $w \cdot x + b$
4. Oh no! The linear combination might not be in $[0,1]$, so we normalize using sigmoid: $\sigma(x) = (1 + e^{-x})^{-1}$
    1. The probability for one class is $\sigma(w \cdot x + b)$, so the other class must have prob $1 - \sigma(w \cdot x + b)$
5. Given our model, we can estimate the probability of a train set under the model $P(\mathscr{D})$
    1. We will set w, b so that $P(\mathscr{D}) = \Pi_i P(c_i \mid d_i)$ is maximal (MLE principle)
    2. For stability and convenience we can take the $\log$ to minimize $-\sum_i \log P(c_i \mid d_i)$ this is CE loss
6. We can then use GD to minimize the CE loss! Since the function is convex, we will converge to the optimum.

Logistic Regression: what's good and what's not

- More freedom in designing features
  - No strong independence assumptions like Naive Bayes
  - Can even have the same feature twice! (*why?*)
- May not work well on small datasets (compared to Naive Bayes)
- Interpreting learned weights can be challenging

PMI: Quick Intuition

PMI: Quick Intuition

$PMI(x, y)$ tells us how correlated two events x, y are:

- $PMI(x, y) = 0$: two events are not correlated at all (if you see x, tells you nothing about y)
- $PMI(x, y) > 0$: two events are correlated (if you see x, you are more likely to see y, vice versa)
- $PMI(x, y) < 0$: two events are anti-correlated (if you see x you are less likely to see y, vice versa)

PMI: Quick Intuition

$PMI(x, y)$ tells us how correlated two events x, y are:

- $PMI(x, y) = 0$: two events are not correlated at all (if you see x, tells you nothing about y)
- $PMI(x, y) > 0$: two events are correlated (if you see x, you are more likely to see y, vice versa)
- $PMI(x, y) < 0$: two events are anti-correlated (if you see x you are less likely to see y, vice versa)

$$\mathrm{PMI}(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)} \qquad \mathrm{PMI}(w = \mathrm{cherry}, c = \mathrm{pie}) = \log_2 \frac{P(w = \mathrm{cherry}, c = \mathrm{pie})}{P(w = \mathrm{cherry})P(c = \mathrm{pie})}$$

PMI: Quick Intuition

$PMI(x, y)$ tells us how correlated two events x, y are:

- $PMI(x, y) = 0$: two events are not correlated at all (if you see x, tells you nothing about y)
- $PMI(x, y) > 0$: two events are correlated (if you see x, you are more likely to see y, vice versa)
- $PMI(x, y) < 0$: two events are anti-correlated (if you see x you are less likely to see y, vice versa)

$$\text{PMI}(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)} \qquad \text{PMI}(w = \text{cherry}, c = \text{pie}) = \log_2 \frac{P(w = \text{cherry}, c = \text{pie})}{P(w = \text{cherry})P(c = \text{pie})}$$

$$\text{PPMI}(w, c) = \max\left(\log_2 \frac{P(w, c)}{P(w)P(c)}, 0\right)$$

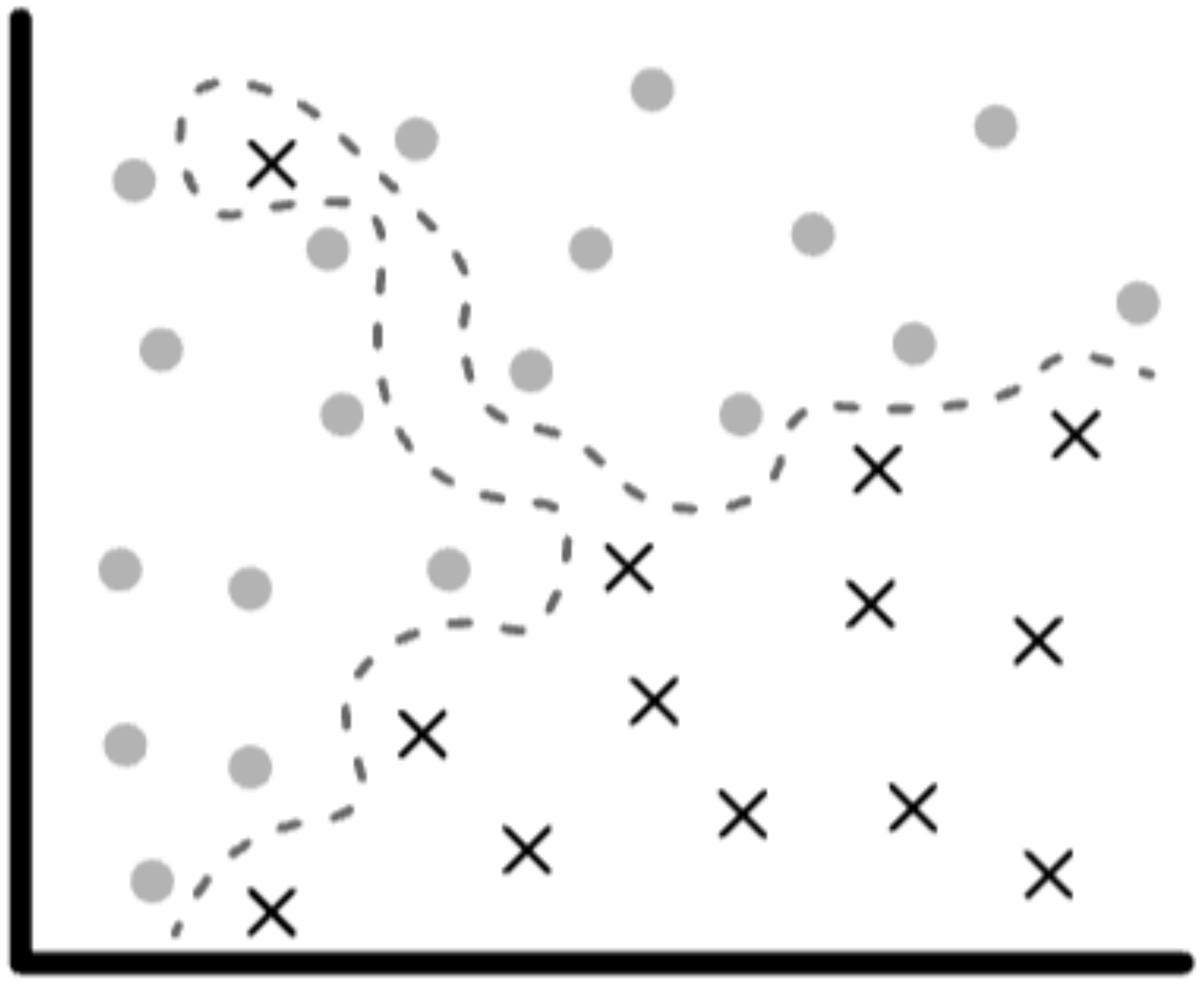
Regularization

Regularization is a technique to help **reduce overfitting**

Regularization

Regularization is a technique to help **reduce overfitting**

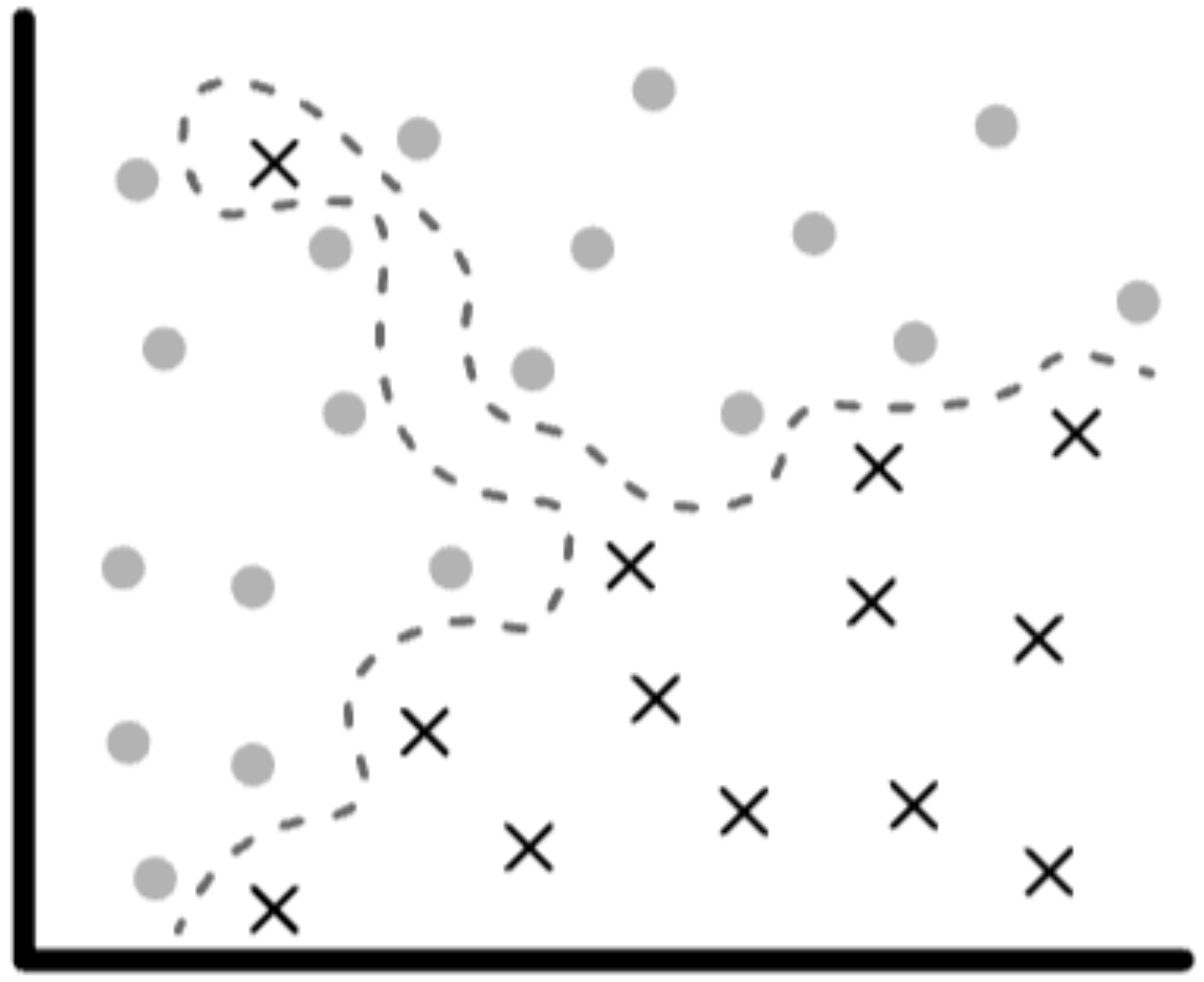
- On the bias vs variance tradeoff scale, **regularization tries to reduce the variance**.



Regularization

Regularization is a technique to help **reduce overfitting**

- On the bias vs variance tradeoff scale, **regularization tries to reduce the variance**. Instead of letting model be anything, we want models where:



Regularization

Regularization is a technique to help **reduce overfitting**

- On the bias vs variance tradeoff scale, **regularization tries to reduce the variance**. Instead of letting model be anything, we want models where:
    - $\|\theta\|^2$ should be small (Ridge regression) (note $\|w\|^2$ in the slides)
    - $\|\theta\|_1$ should be small (Lasso regression)

Regularization

Regularization is a technique to help **reduce overfitting**

- On the bias vs variance tradeoff scale, **regularization tries to reduce the variance**. Instead of letting model be anything, we want models where:
  - $|| \theta ||^2$ should be small (Ridge regression) (note $|| w ||^2$ in the slides)
  - $|| \theta ||_1$ should be small (Lasso regression)

- Practically how do we introduce this to our models? Add it to the loss function! If the original loss function is: L

Regularization

Regularization is a technique to help **reduce overfitting**

- On the bias vs variance tradeoff scale, **regularization tries to reduce the variance**. Instead of letting model be anything, we want models where:
  - $|| \theta ||^2$ should be small (Ridge regression) (note $|| w ||^2$ in the slides)
  - $|| \theta ||_1$ should be small (Lasso regression)

- Practically how do we introduce this to our models? Add it to the loss function! If the original loss function is: L
  - Ridge Regression new loss: $\ell_{\text{ridge}}(\theta) = L(\theta) + \lambda || \theta ||^2$
  - Lasso Regression new loss: $\ell_{\text{lasso}}(\theta) = L(\theta) + \lambda || \theta ||_1$

Regularization

Regularization is a technique to help **reduce overfitting**

- On the bias vs variance tradeoff scale, **regularization tries to reduce the variance**. Instead of letting model be anything, we want models where:
    - $|| \theta ||^2$ should be small (Ridge regression) (note $|| w ||^2$ in the slides)
    - $|| \theta ||_1$ should be small (Lasso regression)

- Practically how do we introduce this to our models? Add it to the loss function! If the original loss function is: L
    - Ridge Regression new loss: $\ell_{\text{ridge}}(\theta) = L(\theta) + \lambda || \theta ||^2$
    - Lasso Regression new loss: $\ell_{\text{lasso}}(\theta) = L(\theta) + \lambda || \theta ||_1$

- The key difference between ridge and lasso regression:
    - Weights in Lasso go to 0, so it can be used for feature selection!