Precept 1

COS 484

Howard Yen
Based on Anika Maskara, Austin Wang, and many previous TAs

Logistics

Precepts: Friday 12pm - 1pm, optional

Friend Center 004

The TA will rotate every week

Goal:

- Review and expand on the topics covered in lecture
- Additional materials

Useful resources

- Working with Colab
- Working with LaTeX
- Submitting Assignments

Feel free to post any issues with any of these on Ed!

Useful resources

- Working with Colab
- Working with LaTeX
- Submitting Assignments

Feel free to post any issues with any of these on Ed!

Classification

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 $P(c \mid d)$

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 $P(c \mid d)$

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 $P(c \mid d)$

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

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

MAP: maximum a posteriori

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 $P(c \mid d)$

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

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

MAP: maximum a posteriori

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

Apply 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 $P(c \mid d)$

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

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

MAP: maximum a posteriori

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

Apply Bayes Rule

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

P(d) does not depend on c!

Naive Bayes: An “illustration”

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

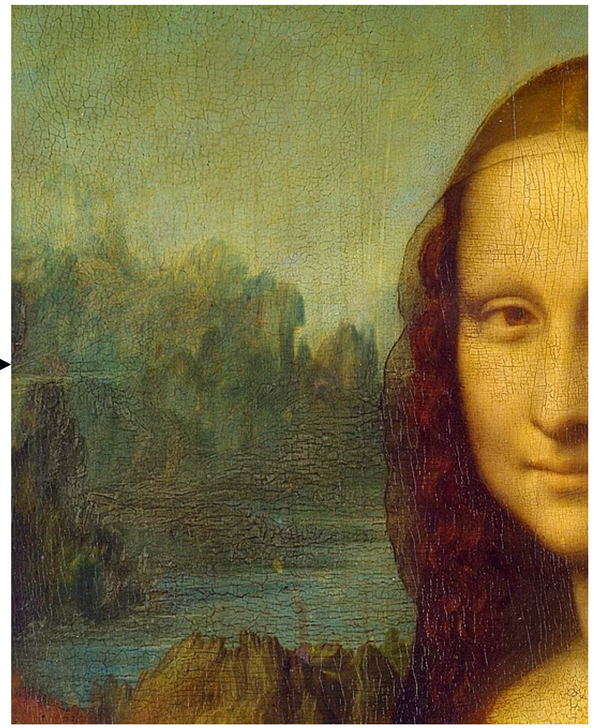
Let’s say you work on a group project with a friend, and we want a model that can attribute your writing vs your friend’s writing. C = {you, friend}

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}

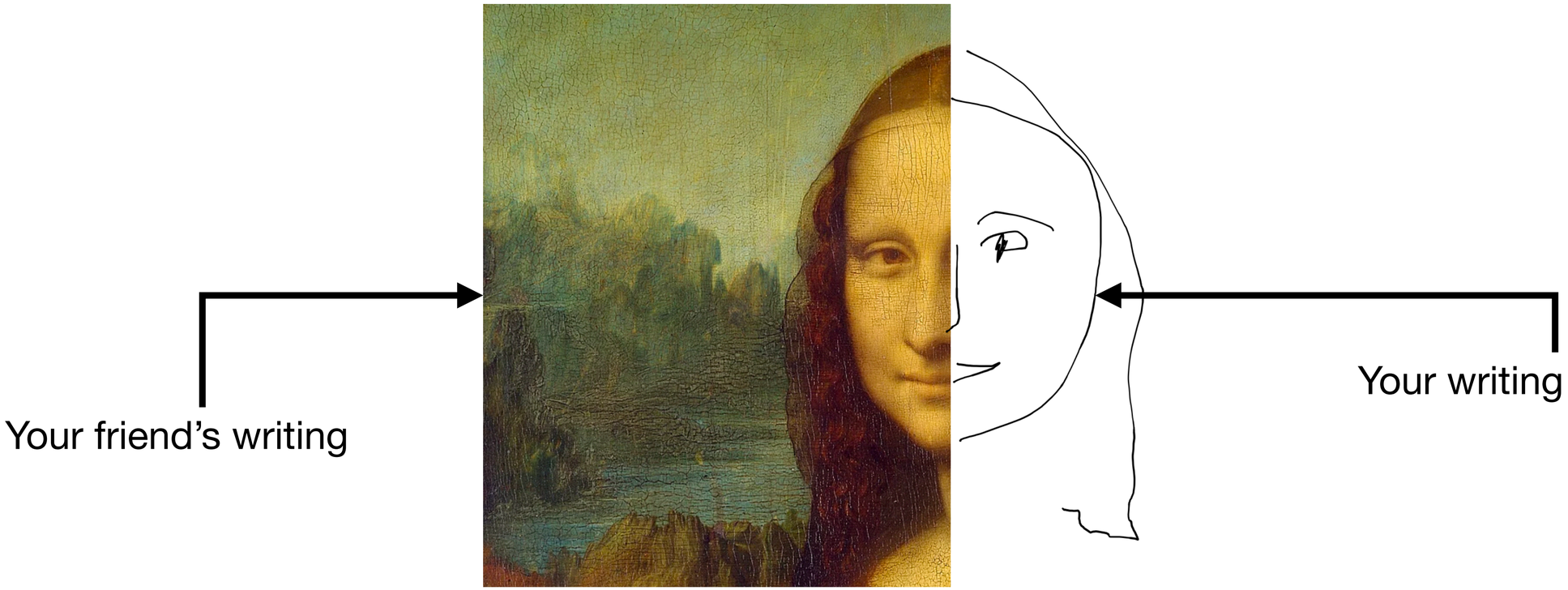
Your friend’s writing



Naive Bayes: An “illustration”

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

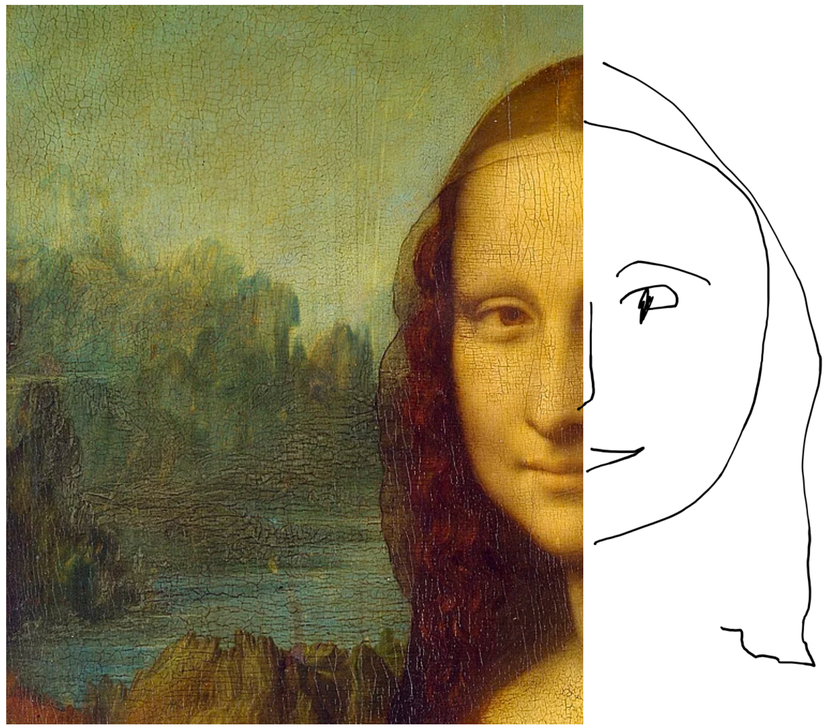
Let’s say you work on a group project with a friend, and we want a model that can attribute your writing vs your friend’s writing. C = {you, friend}



Naive Bayes: An “illustration”

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

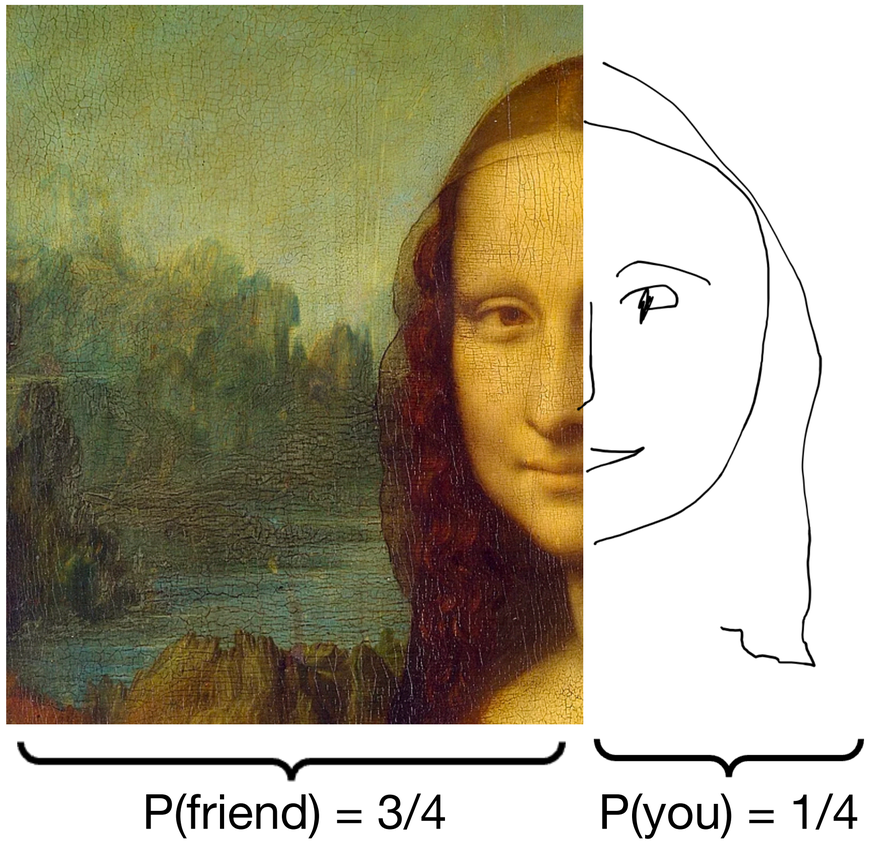
First, we determine who did more work. This gives us a prior estimate (bias) on whether any document was written by you or your friend.



Naive Bayes: An “illustration”

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

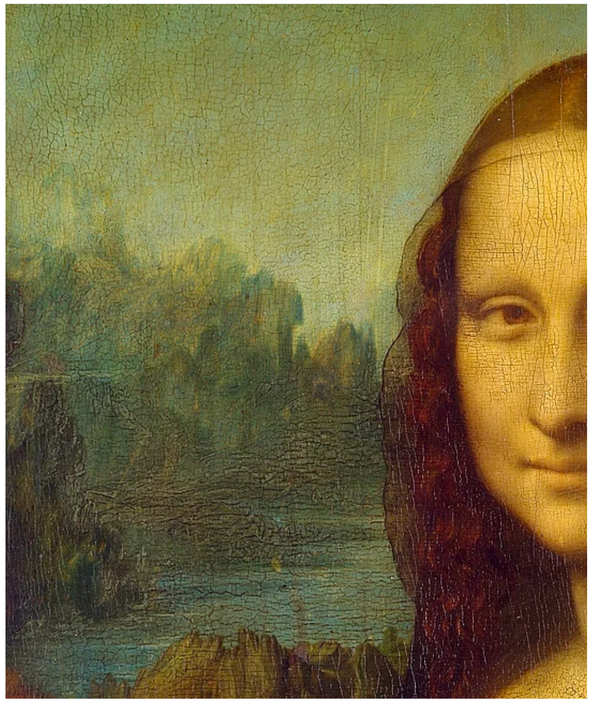
First, we determine who did more work. This gives us a prior estimate (bias) on whether any document was written by you or your friend.



Naive Bayes: An “illustration”

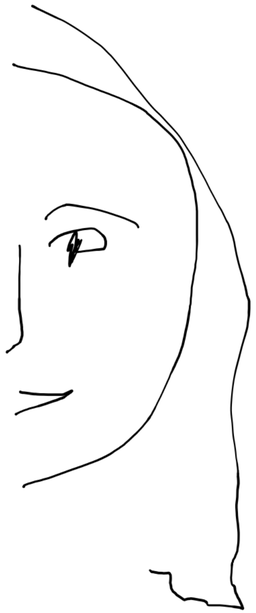
Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c) P(c)$

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|friend)

P(x|you)



Naive Bayes: An “illustration”

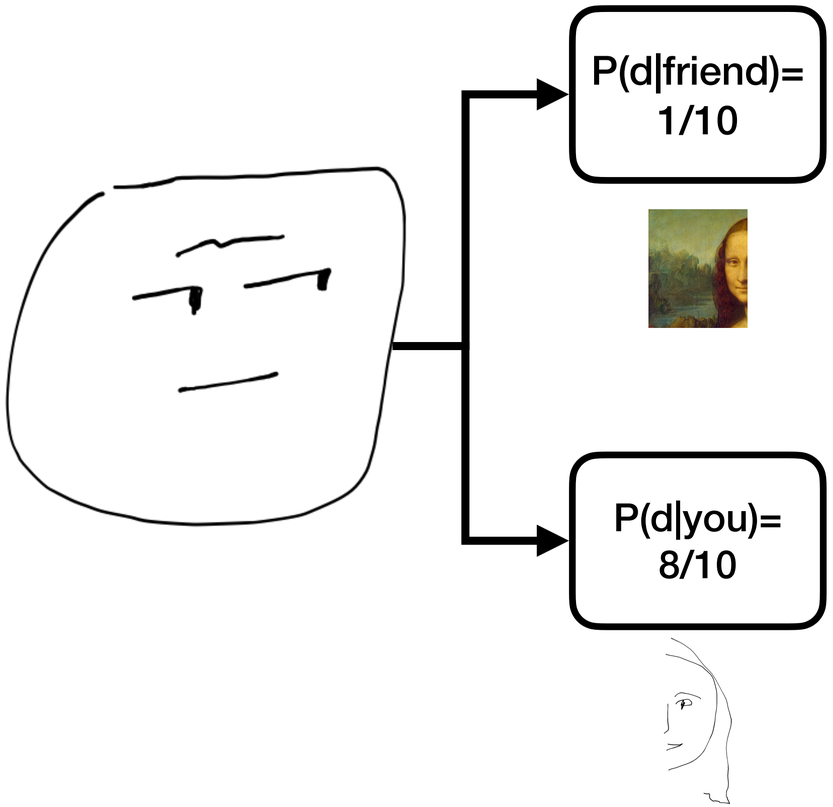
Summary: We want to find the class $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

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 $c_{MAP} = \arg\max_{c \in C} P(d \mid c)P(c)$

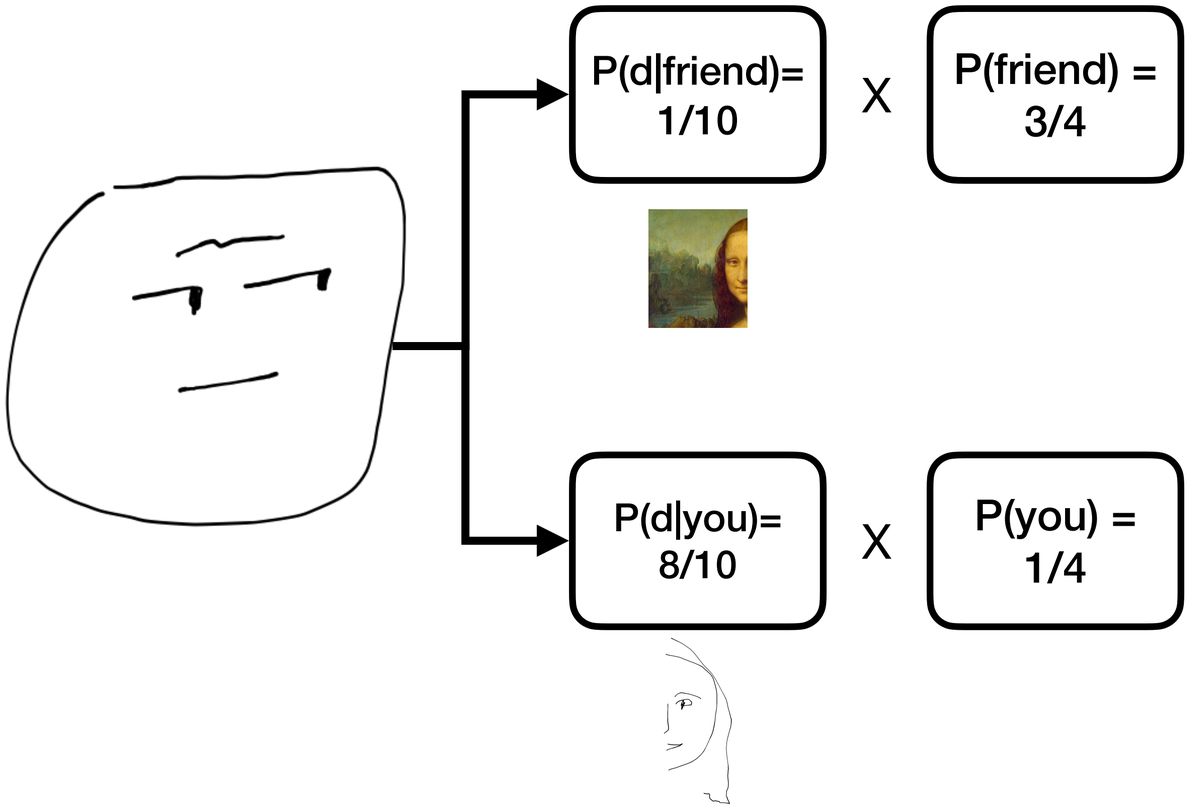
Now given a new sample d, we can compute the probability under each LM to find $P(d \mid c)$. And multiply this by $P(c)$ to find the MAP estimate.



Naive Bayes: An “illustration”

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

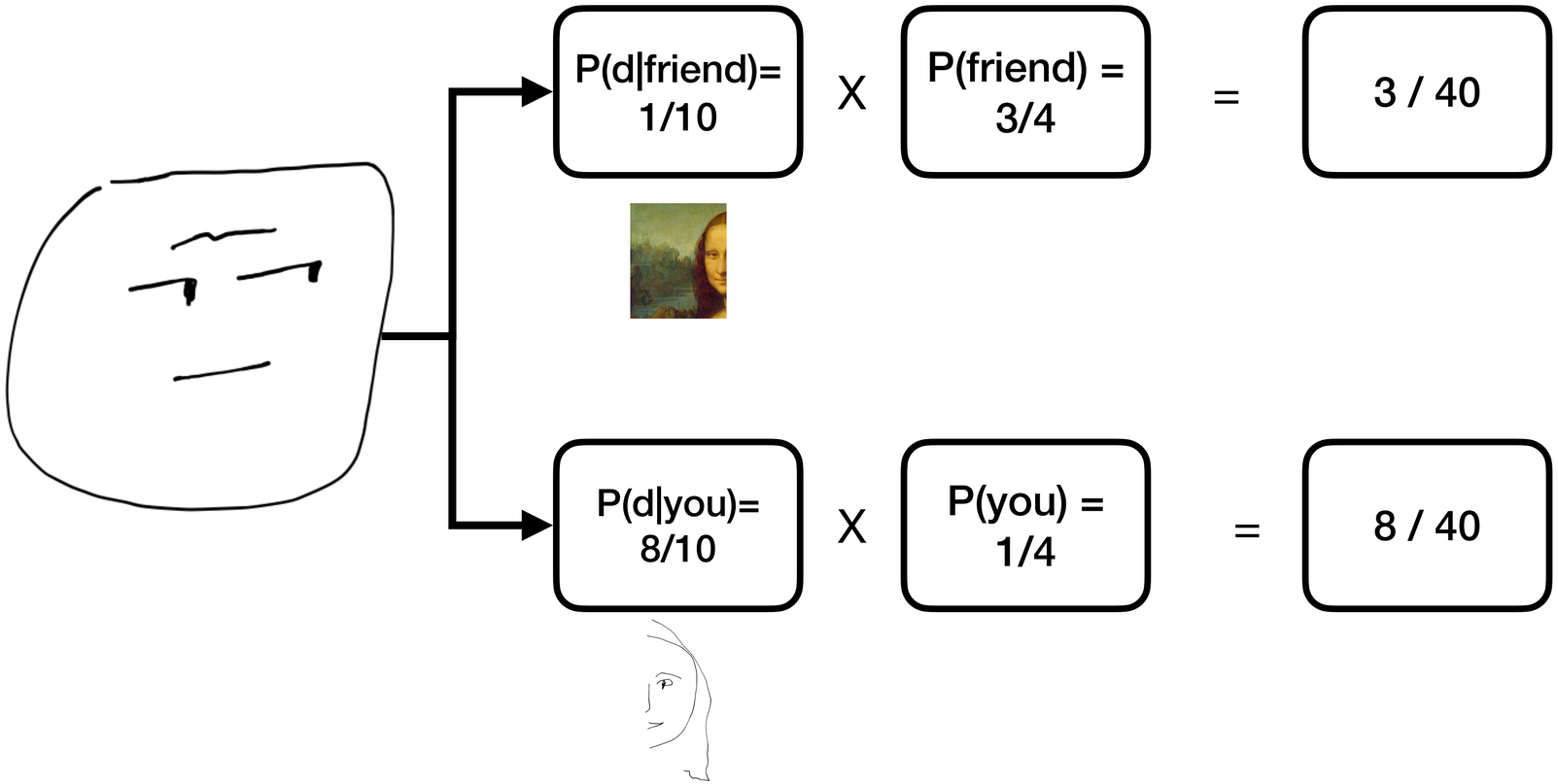
Now given a new sample d, we can compute the probability under each LM to find $P(d \mid c)$. And multiply this by $P(c)$ to find the MAP estimate.



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.

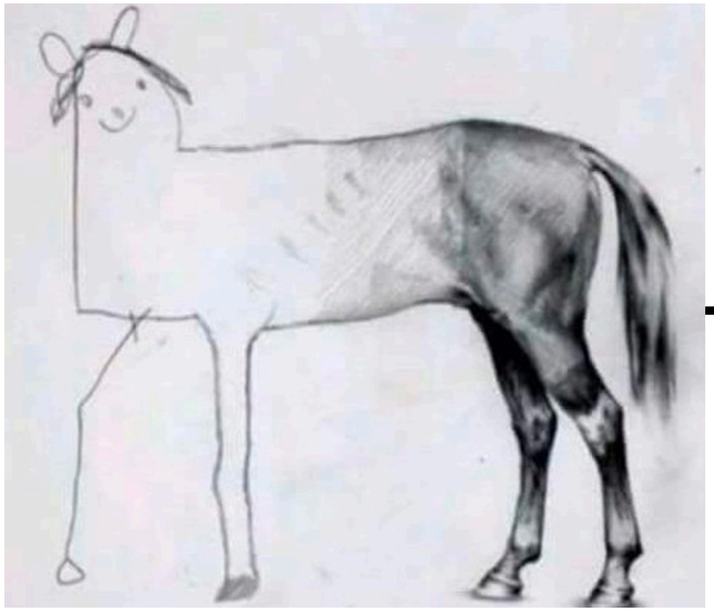


MAP estimate is you!

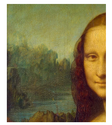
Naive Bayes: An “illustration”

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



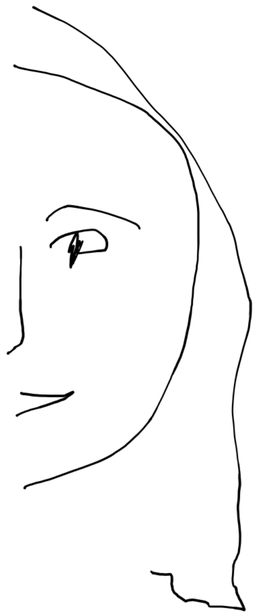
MAP estimate is friend!

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



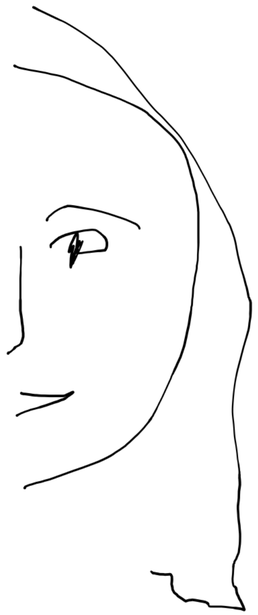
P(x|you)

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

Naive Bayes: One extra detail…

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

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)

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

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
- Very good in domains with many equally important features

  Decision trees suffer from fragmentation in such cases — especially if little data
- 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

Classification

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

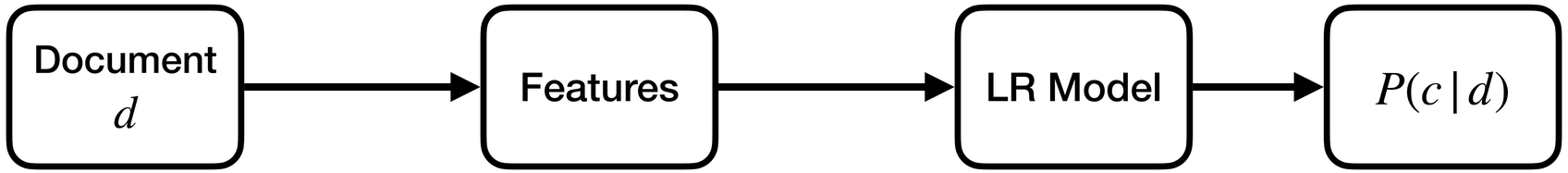
Given a document $d = w_1, \ldots, w_K$ and a set of classes $\mathscr{C} = \{c_1, \ldots, c_m\}$, we want to find the class c_i that maximizes $P(c \mid d)$

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 $P(c \mid d)$

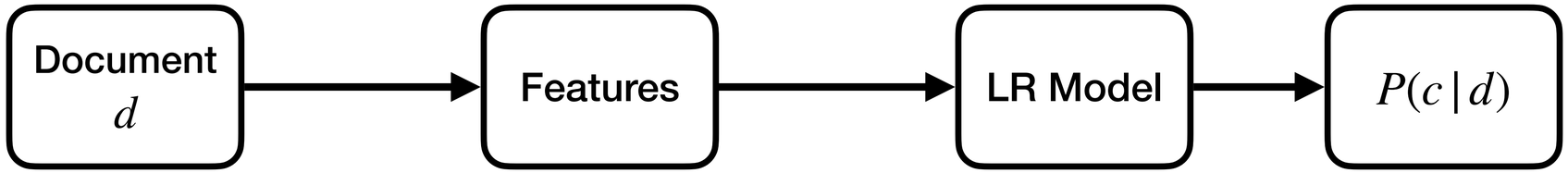
Compared to NB, with LR we take a more direct approach: directly compute $P(c \mid d)$ given a set of features constructed from the input document d.



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.



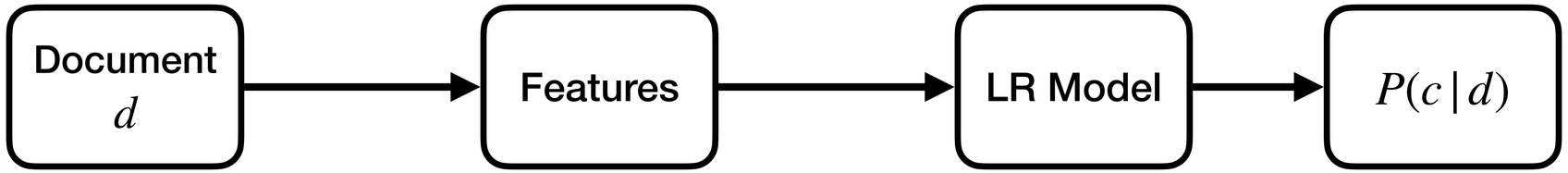
Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise}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



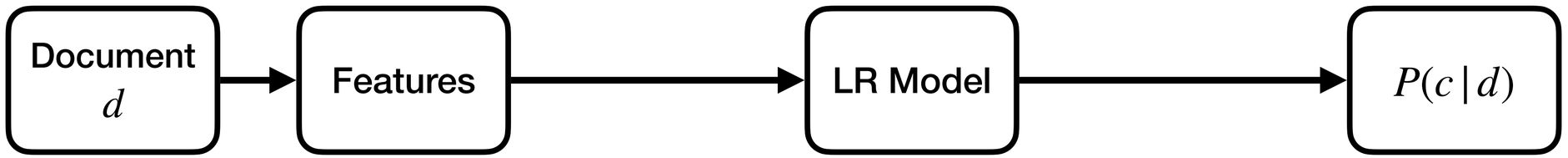
Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	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. Also known as Bag of Words (BOW) model

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?

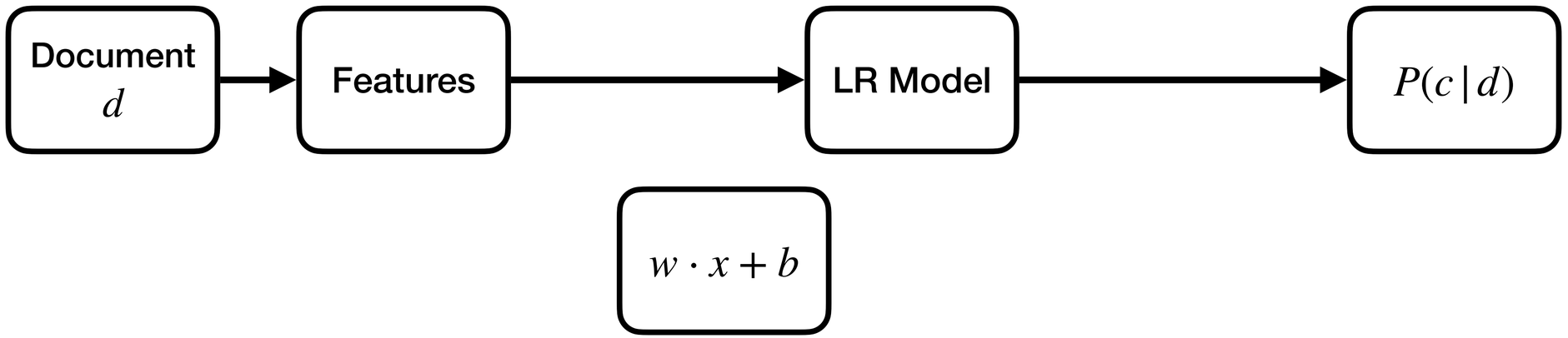


Logistic Regression: LR Model

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.**

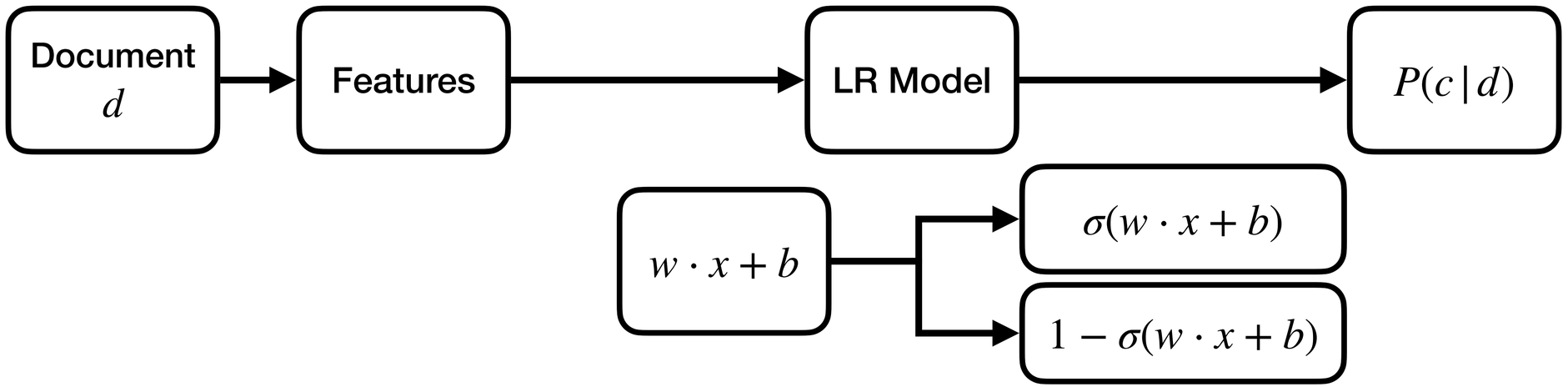


Logistic Regression: LR Model

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 $P(c \mid d)$

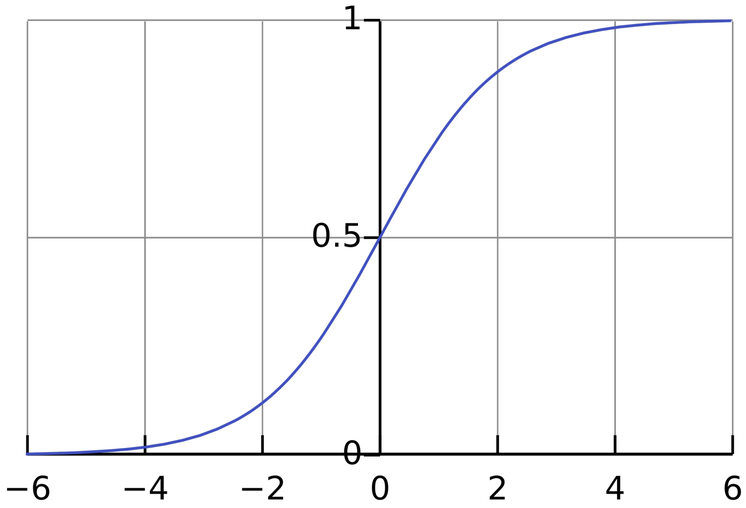
Now given some feature vector x how do we turn this to a probability?

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

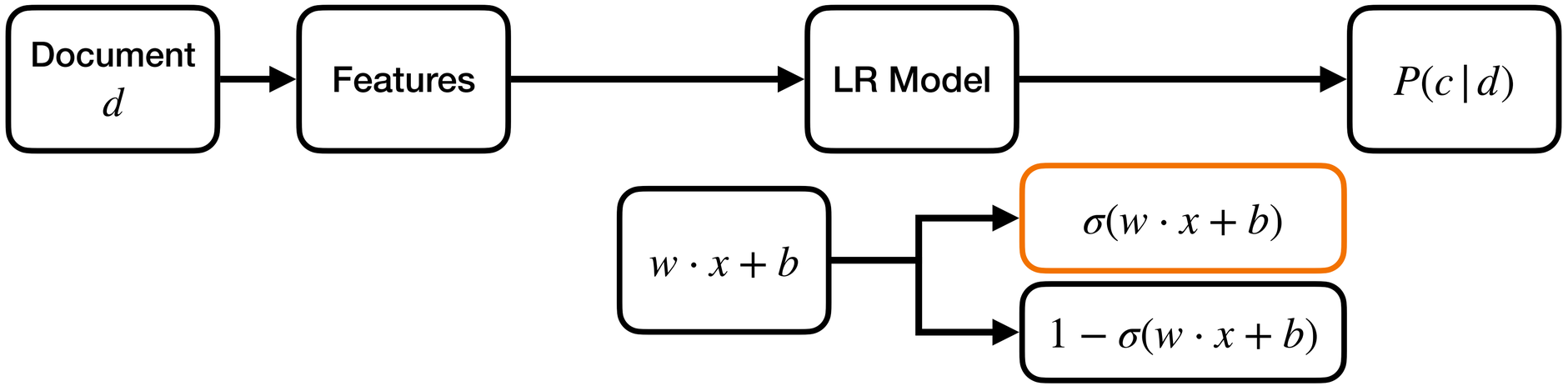


Sigmoid Function $\sigma(x)$

$$\sigma(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$



- A critical piece of logistical regression!
- Turns any arbitrary number into a value between 0 and 1
- Important properties: smooth and, thus, differentiable



Softmax Function

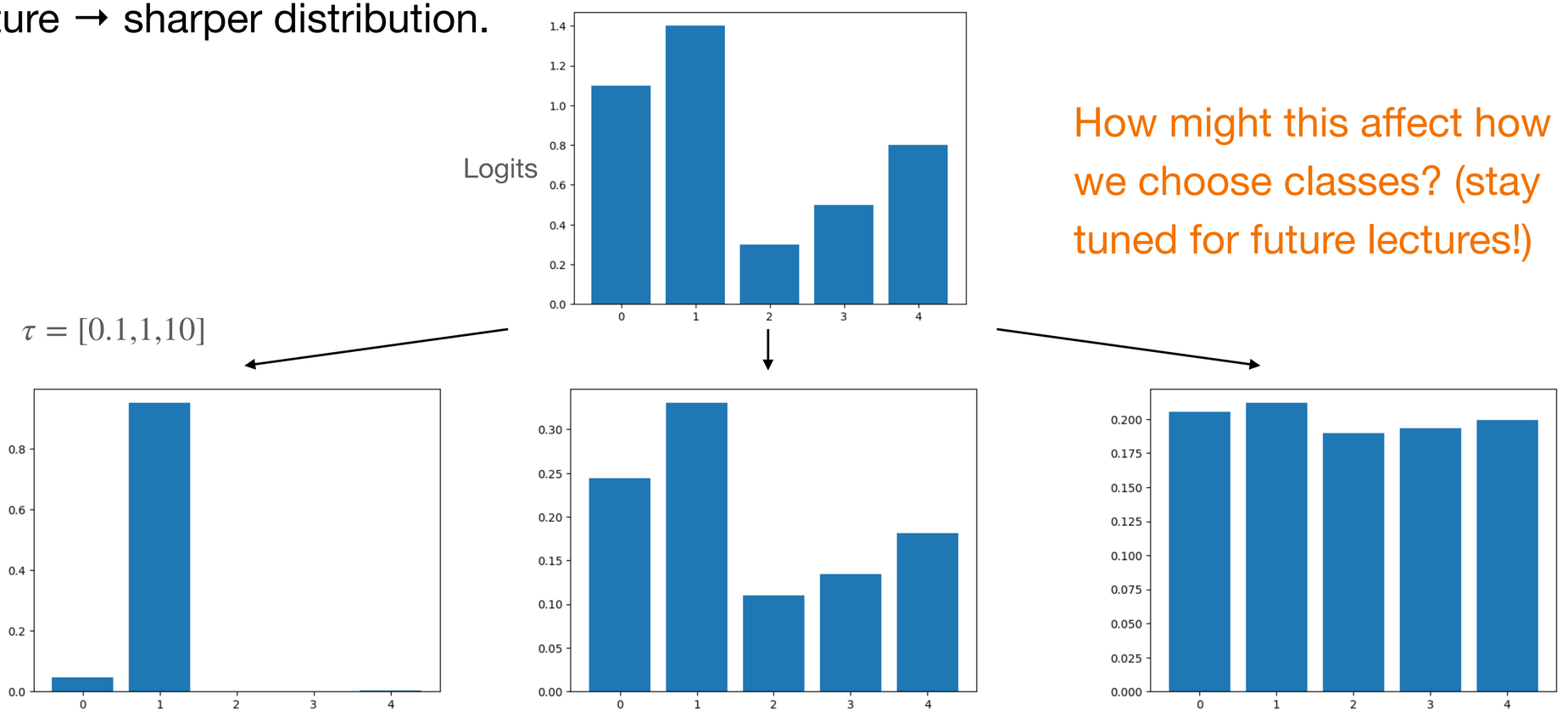
$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, 1 \leq i \leq K$$

- A function for turning a set of numbers into a probability distribution (i.e., values between 0 and 1)
- Useful for classifying a finite number of labels (e.g., words in a vocabulary)
- You may see σ for denoting softmax in some places

Softmax Function: Temperature

$$softmax(x_i) = \frac{e^{x_i/\tau}}{\sum_{j=1}^{K} e^{x_j/\tau}}, 1 \leq i \leq K$$

- Temperature can be used to control the “sharpness” of the probability distribution.
- Lower temperature → sharper distribution.



How might this affect how we choose classes? (stay tuned for future lectures!)

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 $d = w_1, \ldots, w_K$ 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

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
  - More robust to correlated features ("San Francisco" vs "Boston")
    —LR is likely to work better than NB
  - 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

Logistic Regression: Cross Entropy Loss L_{CE}

$$\hat{y}_i = \sigma(\mathbf{w} \cdot \mathbf{x}_i + b)$$

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

Loss function for binary logistic regression

Intuition: we want to estimate how close the classifier output $\hat{y}$ is to the true label y (0 or 1), this allows us to try to **maximize** the **(log) probability** of the true label y for given input x.

Note that this is same as **minimizing** the **negative log probability** or the true label y.

Logistic Regression: Cross Entropy Loss L_{CE}

$$\hat{y}_i = \sigma(\mathbf{w} \cdot \mathbf{x}_i + b)$$

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

Loss function for binary logistic regression

1. how did we arrive at this loss?

For a distribution with discrete outcomes (0 or 1, also known as a Bernoulli distribution), the probability of the correct label is simply:

$$p(y|x) = \hat{y}^y (1 - \hat{y})^{1-y}$$

What does this evaluate to when $y = 0$? What about when $y = 1$?

$$p(y|x) = \hat{y}, y = 1$$

$$p(y|x) = 1 - \hat{y}, y = 0$$

Logistic Regression: Cross Entropy Loss L_{CE}

$$\hat{y}_i = \sigma(\mathbf{w} \cdot \mathbf{x}_i + b)$$

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

Loss function for binary logistic regression

1. how did we arrive at this loss?

For a distribution with discrete outcomes (0 or 1, also known as a Bernoulli distribution), the probability of the correct label is simply:

$$p(y|x) = \hat{y}^y(1 - \hat{y})^{1-y}$$

$$\begin{aligned}\log p(y|x) &= \log[\hat{y}^y(1 - \hat{y})^{1-y}] \\ &= y \log \hat{y} + (1 - y)(1 - \hat{y})\end{aligned}$$

Looks pretty similar to the loss above, which is just the sum over all the log probabilities over n samples.

Logistic Regression: Cross Entropy Loss L_{CE}

$\hat{y}_i = \sigma(\mathbf{w} \cdot \mathbf{x}_i + b), \mathbf{w} = [w_1, \ldots, w_d], \mathbf{x} = [x_1, \ldots, x_d]$

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

Loss function for binary logistic regression

2. how do we use this loss? Gradient descent!

Our objective is to find $\arg\min_{\mathbf{w},b} L_{CE}$, but to do so, we need to find the gradient of the loss with respect to

the weights $\mathbf{w}$ and update them. Consider a simplified version for $n = 1$ without loss of generality.

$$\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$$

We will use the sigmoid derivative later.

Logistic Regression: Cross Entropy Loss L_{CE}

$$\frac{dL_{CE}}{dw_j} = \frac{d}{dw_j} - [y \log \hat{y} + (1-y)\log(1-\hat{y})]$$

$$= \frac{d}{dw_j} - [y \log \sigma(\mathbf{w}\cdot\mathbf{x}+b) + (1-y)\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))]$$

$$= -[\frac{d}{dw_j} y \log \sigma(\mathbf{w}\cdot\mathbf{x}+b) + \frac{d}{dw_j}(1-y)\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))]$$

$$= -\frac{y}{\sigma(\mathbf{w}\cdot\mathbf{x}+b)}\frac{d}{dw_j}\sigma(\mathbf{w}\cdot\mathbf{x}+b) - \frac{1-y}{1-\sigma(\mathbf{w}\cdot\mathbf{x}+b)}\frac{d}{dw_j}(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b)) \quad \text{chain rule on log}$$

$$= -\left[\frac{y}{\sigma(\mathbf{w}\cdot\mathbf{x}+b)} - \frac{1-y}{1-\sigma(\mathbf{w}\cdot\mathbf{x}+b)}\right]\frac{d}{dw_j}\sigma(\mathbf{w}\cdot\mathbf{x}+b) \quad \text{rearrange}$$

$$= -\left[\frac{y-\sigma(\mathbf{w}\cdot\mathbf{x}+b)}{\sigma(\mathbf{w}\cdot\mathbf{x}+b)(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))}\right]\sigma(\mathbf{w}\cdot\mathbf{x}+b)(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))\frac{d(\mathbf{w}\cdot\mathbf{x}+b)}{dw_j} \quad \text{chain rule on sigmoid}$$

$$= -\left[\frac{y-\sigma(\mathbf{w}\cdot\mathbf{x}+b)}{\sigma(\mathbf{w}\cdot\mathbf{x}+b)(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))}\right]\sigma(\mathbf{w}\cdot\mathbf{x}+b)(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))x_j$$

$$= -[y-\sigma(\mathbf{w}\cdot\mathbf{x}+b)]x_j$$

$$= [\sigma(\mathbf{w}\cdot\mathbf{x}+b)-y]x_j$$

Multinomial Logistic Regression: L_{CE}

$$P(y = i \mid x) = softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, 1 \leq i \leq K$$

$$L_{CE} = -\sum_{c=1}^{m} [1\{y = c\} \log P(y = c \mid x)]$$

Loss function for multinomial logistic regression

Once again, we will take the derivative of this with respect to the weights.

Since we are using softmax instead of sigmoid, we need to first derive the derivatives of the softmax.

Multinomial Logistic Regression: L_{CE}

$$P(y = i \mid x) = softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, 1 \leq i \leq K$$

$$L_{CE} = -\sum_{c=1}^{m} [1\{y = c\} \log P(y = c \mid x)]$$

Loss function for multinomial logistic regression

Once again, we will take the derivative of this with respect to the weights.

Since we are using softmax instead of sigmoid, we need to first derive the derivatives of the softmax.

$$P(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

$$\frac{d}{dx_i} P(x_i) = P(x_i)(1 - P(x_i))$$

$$\frac{d}{dx_{i'}} P(x_i) = -P(x_i)P(x_{i'}), i' \neq i$$

Softmax

$$P(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

$$\begin{aligned}
\frac{d}{dx_i} P(x_i) &= \frac{d}{dx_i} \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \\
&= \frac{e^{x_i} \sum_{j=1}^{K} e^{x_j} - e^{x_i} e^{x_i}}{\left(\sum_{j=1}^{K} e^{x_j} \right)^2} \\
&= \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \frac{\sum_{j=1}^{K} e^{x_j} - e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \\
&= P(x_i)(1 - P(x_i))
\end{aligned}$$

$$\begin{aligned}
\frac{d}{dx_{i'}} P(x_i) &= \frac{d}{dx_{i'}} \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, i' \neq i \\
&= e^{x_i} \frac{d}{dx_{i'}} \frac{1}{\sum_{j=1}^{K} e^{x_j}} \\
&= e^{x_i} \frac{e^{x_i'}}{\left(\sum_{j=1}^{K} e^{x_j} \right)^2} \\
&= - P(x_i) P(x_{i'})
\end{aligned}$$

Multinomial Logistic Regression: L_{CE}

$$P(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

$$\frac{d}{dx_i} P(x_i) = P(x_i)(1 - P(x_i))$$

$$\frac{d}{dx_{i'}} P(x_i) = -P(x_i)P(x_{i'}), i' \neq i$$

$$L_{CE} = -\sum_{c=1}^{m} [1\{y = c\} \log P(y = c \mid x)]$$

We'll switch to a notation that is easier for derivation.

$\mathbf{y} = [y_1, \ldots, y_K], y_{i=c} = 1, y_{i \neq c} = 0$ (AKA a one-hot vector) $\hat{y}_i = softmax(\mathbf{w_i} \cdot \mathbf{x} + b_i)$.

$$L_{CE} = -\sum_{i=1}^{K} [y_i \log \hat{y}_i]$$

$L_{CE} = -\log \hat{y}_c$, c is the correct class

Multinomial Logistic Regression: L_{CE}

$\mathbf{y} = [y_1, \ldots, y_K], y_{i=c} = 1, y_{i \neq c} = 0$ $\quad\quad$ $\hat{y}_i = softmax(\mathbf{w_i} \cdot \mathbf{x} + b_i)$

$L_{CE} = -\sum_{i=1}^{K} [y_i \log \hat{y}_i]$ $\quad\quad$ $L_{CE} = -\log \hat{y}_c$, c is the correct class

$$P(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

$$\frac{d}{dx_i} P(x_i) = P(x_i)(1 - P(x_i))$$

$$\frac{d}{dx_{i'}} P(x_i) = -P(x_i)P(x_{i'}), i' \neq i$$

$$\begin{aligned}
\frac{d}{d\mathbf{w_i}} L_{CE} &= -\frac{d}{d\mathbf{w_i}} \log \hat{y}_c \\
&= -\frac{1}{\hat{y}_c} \frac{d}{d\mathbf{w_i}} \hat{y}_c \\
&= -\frac{\hat{y}_c(1 - \hat{y}_c)}{\hat{y}_c} \frac{d}{d\mathbf{w_i}} (\mathbf{w_i} \cdot \mathbf{x} + b_c) && i = c \\
&= -(1 - \hat{y}_c)\mathbf{x} = -(1 - P(y = c \,|\, x))\mathbf{x} && \hat{y}_c = \hat{y}_i \\
&= -\frac{\hat{y}_c \hat{y}_i}{\hat{y}_c} \frac{d}{d\mathbf{w_i}} (\mathbf{w_i} \cdot \mathbf{x} + b_c) && i \neq c \\
&= -(0 - \hat{y}_i)\mathbf{x} = -(1 - P(y = c \,|\, x))\mathbf{x} \\
&= -(y_i - \hat{y}_i)\mathbf{x} = -(y_i - P(y = c \,|\, x))\mathbf{x} && \text{general case}
\end{aligned}$$