Generative Classifiers & Naïve Bayes

Robin Jia
USC CSCI 467, Spring 2024
January 30, 2024

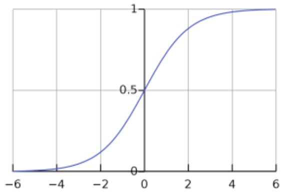
Review: Supervised learning methods (so far)

	Linear Regression	Logistic Regression	Softmax Regression
Task	Regression y is a real number	Binary Classification $y \in \{+1, -1\}$	Multiclass classification $y \in \{1, 2, \ldots, C\}$

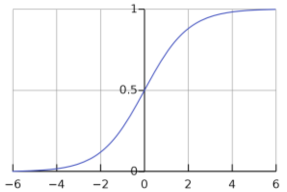
Review: Supervised learning methods (so far)

	Linear Regression	Logistic Regression	Softmax Regression
Task	<u>Regression</u> y is a real number	<u>Binary Classification</u> $y \in \{+1, -1\}$	<u>Multiclass classification</u> $y \in \{1, 2, \ldots, C\}$
Parameters (what to learn)	$w \in \mathbb{R}^d$ (d is dimension of x)	$w \in \mathbb{R}^d$	$w^{(1)}, \ldots, w^{(C)} \in \mathbb{R}^d$ (C*d total params)

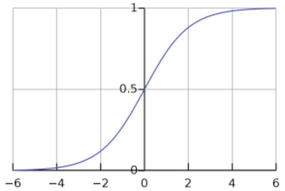
Review: Supervised learning methods (so far)

	Linear Regressn	Logistic Regression	Softmax Regression
Task	Regression y is a real number	Binary Classification $y \in \{+1, -1\}$	Multiclass classification $y \in \{1, 2, \ldots, C\}$
Parameters (what to learn)	$w \in \mathbb{R}^d$ (d is dimension of x)	$w \in \mathbb{R}^d$	$w^{(1)}, \ldots, w^{(C)} \in \mathbb{R}^d$ (C*d total params)
Probabilistic Story (how did nature create the data?)	$y \sim \text{Normal}(w^\top x, \sigma^2)$ Mean (Depends on w) Variance (constant)	$p(y = 1 \mid x) = \sigma(w^\top x)$ Plot of $\sigma(z)$ vs. z 	$p(y = j \mid x) = \frac{\exp(w^{(j)\top} x)}{\boxed{\sum_{k=1}^{C} \exp(w^{(k)\top} x)}}$ Normalizes to probability distribution

Review: Supervised learning methods (so far)

	Linear Regression	Logistic Regression	Softmax Regression
Task	<u>Regression</u> y is a real number	<u>Binary Classification</u> $y \in \{+1, -1\}$	<u>Multiclass classification</u> $y \in \{1, 2, \ldots, C\}$
Parameters (what to learn)	$w \in \mathbb{R}^d$ (d is dimension of x)	$w \in \mathbb{R}^d$	$w^{(1)}, \ldots, w^{(C)} \in \mathbb{R}^d$ (C*d total params)
Probabilistic Story (how did nature create the data?)	$y \sim \text{Normal}(w^\top x, \sigma^2)$ Mean (Depends on w) Variance (constant)	$p(y = 1 \mid x) = \sigma(w^\top x)$ Plot of σ(z) vs. z 	$p(y = j \mid x) = \frac{\exp(w^{(j)\top} x)}{\boxed{\sum_{k=1}^{C} \exp(w^{(k)\top} x)}}$ Normalizes to probability distribution
Loss function (measures how bad any choice of parameters is)	Derive using <u>Principle of Maximum Likelihood Estimation (MLE)</u> Want to maximize probability of data = $\prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; w)$ Same as minimizing negative log-likelihood = $\sum_{i=1}^{n} -\log p(y^{(i)} \mid x^{(i)}; w)$		

Review: Supervised learning methods (so far)

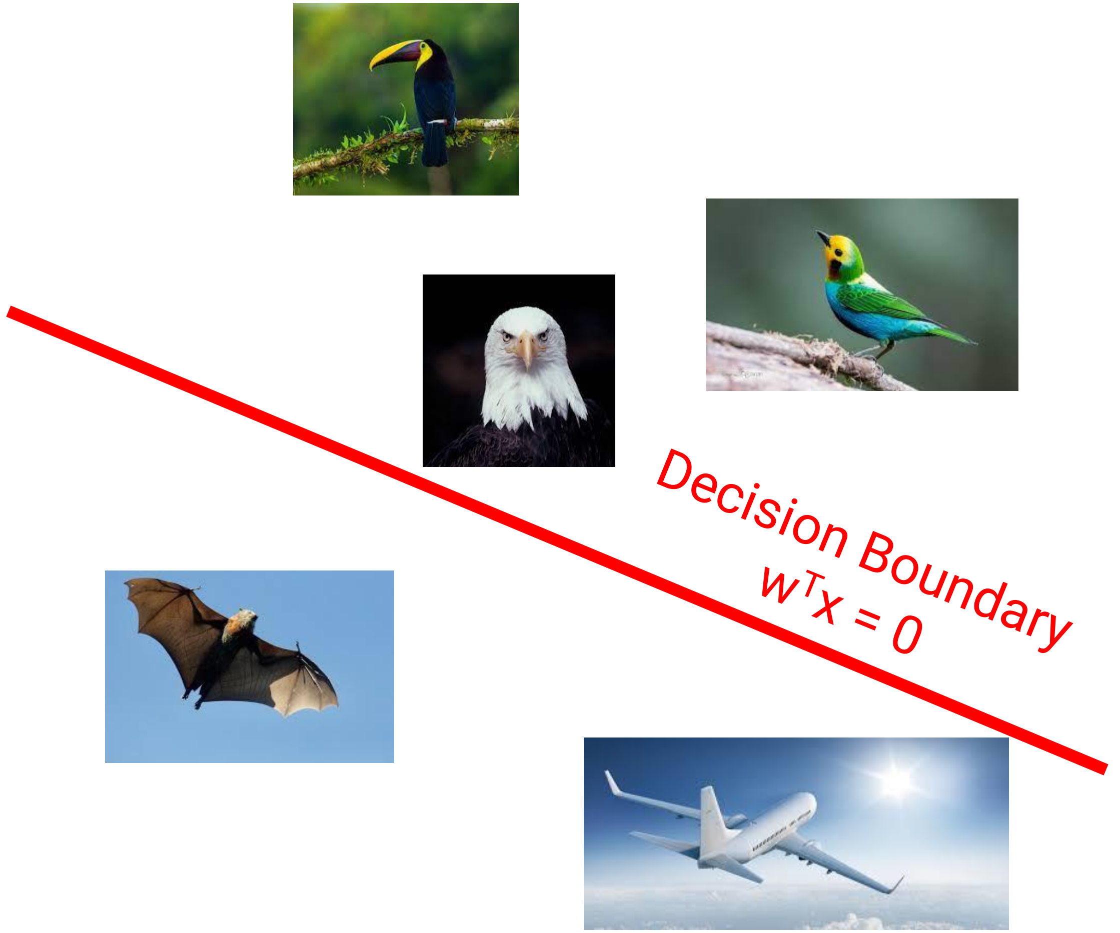
	Linear Regression	Logistic Regression	Softmax Regression
Task	<u>Regression</u> y is a real number	<u>Binary Classification</u> $y \in \{+1, -1\}$	<u>Multiclass classification</u> $y \in \{1, 2, \ldots, C\}$
Parameters (what to learn)	$w \in \mathbb{R}^d$ (d is dimension of x)	$w \in \mathbb{R}^d$	$w^{(1)}, \ldots, w^{(C)} \in \mathbb{R}^d$ (C*d total params)
Probabilistic Story (how did nature create the data?)	$y \sim \text{Normal}(w^T x, \sigma^2)$ Mean (Depends on w) Variance (constant)	$p(y = 1 \mid x) = \sigma(w^\top x)$ Plot of $\sigma(z)$ vs. z 	$p(y = j \mid x) = \dfrac{\exp(w^{(j)\top} x)}{\boxed{\sum_{k=1}^{C} \exp(w^{(k)\top} x)}}$ Normalizes to probability distribution
Loss function (measures how bad any choice of parameters is)	Derive using <u>Principle of Maximum Likelihood Estimation (MLE)</u> Want to maximize probability of data = $\prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; w)$ Same as minimizing negative log-likelihood = $\sum_{i=1}^{n} -\log p(y^{(i)} \mid x^{(i)}; w)$		
How to minimize loss	Gradient Descent or Normal Equations	Gradient Descent	

Today’s Plan

- Generative vs. Discriminative Classifiers
- Naïve Bayes for Text Classification
  - First Attempt
  - Two fixes to avoid zeroes
- Naïve Bayes for Feature Vectors

Discriminative Classifiers

- Train a model with parameters w to model *p(y|x)*
  - Logistic regression: $p(y=1|x) = \sigma(w^T x)$
- Note: We **do not** attempt to model *p(x)*!
  - **Given** an image *x*, classifier predicts whether it is a bird or not
  - Model does not try to describe what an image of a bird actually is
  - Only has to find some features that **discriminate** between birds and non-birds
- Methods like logistic regression & softmax regression are called "**discriminative classifiers**"



Today: Generative classifiers

- Instead of modeling $p(y|x)$, **model the entire joint distribution** $p(x, y)$ as the product $p(y) * p(x|y)$
  - $p(y)$: How often does each label occur? **Easy**
  - $p(x|y)$: What is the space of all possible x's that have the label y? **Can be complex**

Prior: 25% of images are birds

If y=bird,
all possible x's include…



If y=not bird,
all possible x's include…


Predicting with a Generative Classifier

- Suppose we have adequately learned *p(y)* and *p(x|y)*
- At test time, we get an input *x*
- How to predict? **Bayes Rule**

Prediction of label given input

Model estimates these

$$\boxed{p(y \mid x)} = \frac{\boxed{p(y)p(x \mid y)}}{\boxed{p(x)}}$$

Just for normalization

$$\boxed{p(x)} = \sum_j p(y = j)p(x \mid y = j)$$

Prior: 25% of images are birds

If y=bird, all possible x's include…

If y=not bird, all possible x's include…

Test input

Today’s Plan

- Generative vs. Discriminative Classifiers
- Naïve Bayes for Text Classification
  - First Attempt
  - Two fixes to avoid zeroes
- Naïve Bayes for Feature Vectors

Setting: Text Classification

- Each input x is a document
  - Documents can have different numbers of words
  - $x^{(i)}_j$ is j-th word of i-th training example
- Each training example has corresponding label y

Training Data (sentiment analysis)

i	$y^{(i)}$	$x^{(i)}$
1	+1	great acting and score
2	-1	terrible directing
3	+1	great movie
4	-1	terrible
5	+1	amazing

Test Data

x^{test} = "great directing"

Training a generative classifier

- We have to model two things
  - p(y): For each label y, what is the probability of y occurring?
  - p(x|y): For each label y, what corresponding x's are likely to appear?

Training Data

i	$y^{(i)}$	$x^{(i)}$
1	+1	great acting and score
2	-1	terrible directing
3	+1	great movie
4	-1	terrible
5	+1	amazing

Modeling p(y)

- Modeling p(y) is easy: Just count how often each *y* appears!
- Let *C* be the number of possible classes
- Our model learns model parameter $\pi_j = P(y=j)$ for each possible *j*
- Learning: $\pi_j = count(y=j)/n$
  - *count(y=j)*: how often y=j in training data
  - *n*: number of training examples
  - Justification: Maximum likelihood estimate (same as HW0 coin flip problem)

Training Data

i	$y^{(i)}$	$x^{(i)}$
1	+1	great acting and score
2	-1	terrible directing
3	+1	great movie
4	-1	terrible
5	+1	amazing

In this dataset: $y \in \{+1, -1\}$ so **C=2**

5 training examples, so **n=5**
y=+1 occurs 3 times, so $\pi_1 = 3/5 = 0.6$
y=-1 occurs 2 times, so $\pi_{-1} = 2/5 = 0.4$

Training a generative classifier

- We have to model two things
  - p(y): For each label y, what is the probability of y occurring?
  - **p(x|y): For each label y, what corresponding x's are likely to appear?**
    - This is much harder because x's are usually very complex objects
    - Different generative classification methods do different things
    - Today: **Naïve Bayes method**

Training Data

i	$y^{(i)}$	$x^{(i)}$
1	+1	great acting and score
2	-1	terrible directing
3	+1	great movie
4	-1	terrible
5	+1	amazing

Modeling p(x|y) with Naïve Bayes

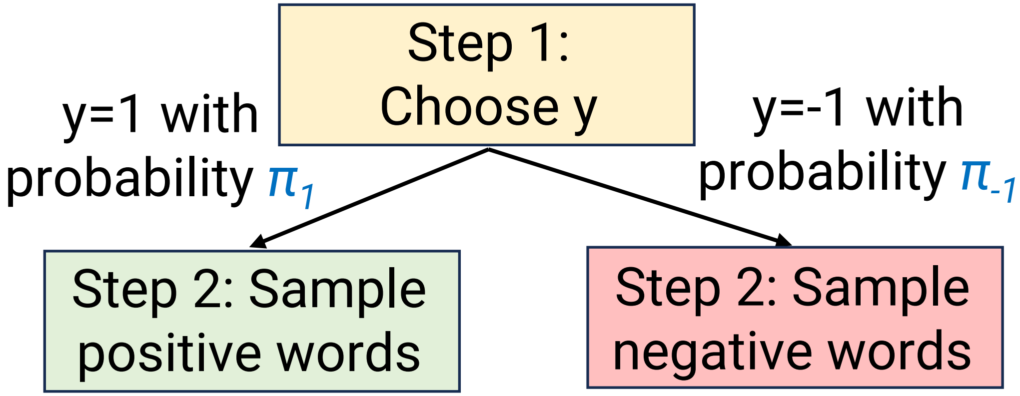
- Idea: Make a simplifying assumption about *p(x|y)* to make it possible to estimate
- Naïve Bayes assumption: Each word of the document *x* is **conditionally independent** given label *y*:

$$p(x \mid y) = \prod_{j=1}^{d} p(x_j \mid y)$$

  - “Once label is chosen, each word is sampled independently”
  - Note: This assumption does not have to be true (it definitely isn’t), just has to be “close enough” so that classifier makes reasonable predictions

The Naïve Bayes Assumption

- Naïve Bayes posits its own **probabilistic story** about how the data was generated
- Step 1: Each $y^{(i)}$ was sampled from the prior distribution $p(y)$
  - “First, decide to either write a positive or negative review”
- Step 2: Each word in $x^{(i)}$ was sampled **independently** from the word distribution for label $y^{(i)}$
  - “If you decided to be positive, write the document by randomly sampling positive-sounding words”
  - “If you decided to be negative, write the document by randomly sampling negative-sounding words”
  - Each word is **independent when conditioning on *y***
- Models the entire process of generating x and y



P(word|y=1)	word
0.0050	great
0.0042	the
0.0035	good
0.0032	movie
…	…

“movie good the great score…”

P(word|y=-1)	word
0.0054	bad
0.0045	movie
0.0041	worst
0.0034	is
…	…

“worst acting is movie bad…”

Why is the Naïve Bayes Assumption OK?

- Clearly, documents generated in this way don't look very realistic!
- Why is this OK?
  - We don't need our p(x|y) to actually generate good documents
  - We just need it to be reasonable enough so that when given a real document *x*,

    p(x|true y) > p(x|other y)
  - Can be bad at modeling all the complex things that aren't related to *y* (grammar, writing style, etc.)

Step 1: Choose y

y=1 with probability π_1 → Step 2: Sample positive words

y=-1 with probability π_{-1} → Step 2: Sample negative words

P(word|y=1)	word
0.0050	great
0.0042	the
0.0035	good
0.0032	movie
…	…

"movie good the great score…"

P(word|y=-1)	word
0.0054	bad
0.0045	movie
0.0041	is
0.0034	worst
…	…

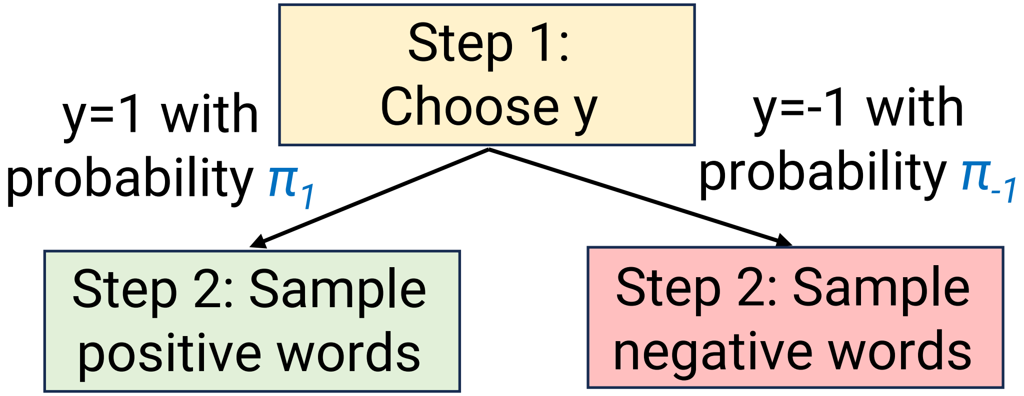
"worst acting is movie bad…"

Learning with Naïve Bayes

- Let *V* ("vocabulary") denote the set of words in the dictionary
- Model learns parameter $\tau_{wj} = P(w|y=j)$
  - For each word *w* in *V*
  - For each possible label *j*
  - Total of $|V| * C$ parameters to learn
- How to learn? Just count!
  - For each word w and label j, learn:

    $$\tau_{wj} = \frac{[\#occurrences\ of\ w\ when\ y=j]}{[total\ words\ when\ y=j]}$$

  - Again justified by MLE
  - Note: This formula has a flaw, which we will fix later



P(word|y=1)	word
???	a
???	ab
???	abacus
???	abate
…	…

P(word|y=-1)	word
???	a
???	ab
???	abacus
???	abate
…	…

Learning goal: Estimate all the ???'s

Learning with Naïve Bayes

Training Data

i	$y^{(i)}$	$x^{(i)}$
1	+1	$\boxed{\text{great}}$ $\boxed{\text{acting}}$ $\boxed{\text{and}}$ $\boxed{\text{score}}$
2	-1	terrible directing
3	+1	$\boxed{\text{great}}$ $\boxed{\text{movie}}$
4	-1	terrible
5	+1	$\boxed{\text{amazing}}$

- For each of y=+1 and y=-1, want to learn a distribution over 8 words
- 7 total words appear with y=+1

Parameters to learn

$\tau_{w,1}$	word w
???	acting
???	and
???	amazing
???	directing
???	great
???	movie
???	score
???	terrible

$\tau_{w,-1}$	word w
???	acting
???	and
???	amazing
???	directing
???	great
???	movie
???	score
???	terrible

Learning with Naïve Bayes

Training Data

i	$y^{(i)}$	$x^{(i)}$
1	+1	great acting and score
2	-1	terrible directing
3	+1	great movie
4	-1	terrible
5	+1	amazing

- For each of y=+1 and y=-1, want to learn a distribution over 8 words
- 7 total words appear with y=+1
- Count each word and divide by total

Parameters to learn

$\tau_{w,1}$	word w
1/7	acting
1/7	and
1/7	amazing
0	directing
2/7	great
1/7	movie
1/7	score
0	terrible

$\tau_{w,-1}$	word w
???	acting
???	and
???	amazing
???	directing
???	great
???	movie
???	score
???	terrible

Learning with Naïve Bayes

Training Data

i	$y^{(i)}$	$x^{(i)}$
1	+1	great acting and score
2	-1	terrible directing
3	+1	great movie
4	-1	terrible
5	+1	amazing

- For each of y=+1 and y=-1, want to learn a distribution over 8 words
- 7 total words appear with y=+1
- Count each word and divide by total
- Repeat for y=-1 (3 total words)

Parameters to learn

$\tau_{w,1}$	word w
1/7	acting
1/7	and
1/7	amazing
0	directing
2/7	great
1/7	movie
1/7	score
0	terrible

$\tau_{w,-1}$	word w
0	acting
0	and
0	amazing
1/3	directing
0	great
0	movie
0	score
2/3	terrible

Predicting with Naïve Bayes

- Given test example x^{test} = "great score"
- Compute p(x, y=+1)
  = p(y=+1) * p(x|y=+1)
  = p(y=+1) * p("great"|y=+1) * p("score"|y=+1)
  = 3/5 * 2/7 * 1/7 = **0.0245**
- Compute p(x, y=-1)
  = p(y=-1) * p(x|y=-1)
  = p(y=-1) * p("great"|y=-1) * p("score"|y=-1)
  = 2/5 * 0 * 0 = **0**
- By Bayes Rule:
  - P(y=+1|x) = 0.0245/(0.0245+0) = 1
  - Model is sure that y=+1, so predict +1
  - Always predict y with largest p(x, y)

Learned Parameters

$\pi_1 = 3/5$

$\tau_{w,1}$	word w
1/7	acting
1/7	and
1/7	amazing
0	directing
2/7	**great**
1/7	movie
1/7	**score**
0	terrible

$\pi_{-1} = 2/5$

$\tau_{w,-1}$	word w
0	acting
0	and
0	amazing
1/3	directing
0	**great**
0	movie
0	**score**
2/3	terrible

Announcements

- HW1 out, due next Tuesday
- HW0 grades returned
  - Regrades will be open for 1 more week
  - Check blackboard for solutions before asking for regrade
  - In general: will keep regrades open for 1 week after returning grades
- Friday section: Follow-up to last Thursday's class
  - Cross-validation: Another way to evaluate on held-out data
  - Choosing an appropriate evaluation metric

Today’s Plan

- Generative vs. Discriminative Classifiers
- Naïve Bayes for Text Classification
  - First Attempt
  - Two fixes to avoid zeroes
- Naïve Bayes for Feature Vectors

Problem #1: Too Many Zeroes

- Given test example x^{test} = "great directing"
- Compute p(x, y=+1)
  = p(y=+1) * p(x|y=+1)
  = p(y=+1) * p("great"|y=+1) * p("directing"|y=+1)
  = 3/5 * 2/7 * **0** = **0**
- Compute p(x, y=-1)
  = p(y=-1) * p(x|y=-1)
  = p(y=-1) * p("great"|y=-1) * p("directing"|y=-1)
  = 2/5 * **0** * 1/3 = **0**
- By Bayes Rule:
  - P(y=+1|x) = 0/(0+0) = **NaN**
  - **Model thinks this x^{test} is impossible!**

Learned Parameters

$\pi_1 = 3/5$

$\tau_{w,1}$	word w
1/7	acting
1/7	and
1/7	amazing
0	**directing**
2/7	**great**
1/7	movie
1/7	score
0	terrible

$\pi_{-1} = 2/5$

$\tau_{w,-1}$	word w
0	acting
0	and
0	amazing
1/3	**directing**
0	**great**
0	movie
0	score
2/3	terrible

Avoiding Zeroes with Laplace Smoothing

- Problem : Assign probability of 0 to many (word, label) pairs
- Solution: **Laplace Smoothing**
  - Imagine that every (word, label) pair was seem an additional λ times
    - λ is a new hyperparameter
  - New formula:

$$\tau_{wy} = \frac{[\#occurrences\ of\ w\ when\ y=j] + \boldsymbol{\lambda}}{[total\ words\ when\ y=j] + \boxed{|\boldsymbol{V}| * \boldsymbol{\lambda}}}$$

Add λ for each word in V,
so total # of imaginary counts is |V| * λ

Parameters to learn

$\tau_{w,1}$	word w
1/7	acting
1/7	and
1/7	amazing
0	directing
2/7	great
1/7	movie
1/7	score
0	terrible

$\tau_{w,-1}$	word w
0	acting
0	and
0	amazing
1/3	directing
0	great
0	movie
0	score
2/3	terrible

Laplace Smoothing Example

Training Data

i	$y^{(i)}$	$x^{(i)}$
1	+1	great acting and score
2	-1	terrible directing
3	+1	great movie
4	-1	terrible
5	+1	amazing

Parameters to learn

$\tau_{w,1}$	word w
1/7	acting
1/7	and
1/7	amazing
0	directing
2/7	great
1/7	movie
1/7	score
0	terrible

$\tau_{w,-1}$	word w
0	acting
0	and
0	amazing
1/3	directing
0	great
0	movie
0	score
2/3	terrible

With no Laplace Smoothing

Laplace Smoothing Example

Training Data

i	$y^{(i)}$	$x^{(i)}$
1	+1	great acting and score
2	-1	terrible directing
3	+1	great movie
4	-1	terrible
5	+1	amazing

$$\tau_{wy} = \frac{\textit{[\#occurrences of w when y=j]} + \boldsymbol{\lambda}}{\textit{[total words when y=j]} + |\mathbf{V}| * \boldsymbol{\lambda}}$$

Parameters to learn

$\tau_{w,1}$	word w
(1+1)/(7+8)	acting
(1+1)/(7+8)	and
(1+1)/(7+8)	amazing
(0+1)/(7+8)	directing
(2+1)/(7+8)	great
(1+1)/(7+8)	movie
(1+1)/(7+8)	score
(0+1)/(7+8)	terrible

$\tau_{w,-1}$	word w
(0+1)/(3+8)	acting
(0+1)/(3+8)	and
(0+1)/(3+8)	amazing
(1+1)/(3+8)	directing
(0+1)/(3+8)	great
(0+1)/(3+8)	movie
(0+1)/(3+8)	score
(2+1)/(3+8)	terrible

Laplace Smoothing with λ = 1

Laplace Smoothing Example

Training Data

i	$y^{(i)}$	$x^{(i)}$
1	+1	great acting and score
2	-1	terrible directing
3	+1	great movie
4	-1	terrible
5	+1	amazing

$$\tau_{wy} = \frac{[\#occurrences\ of\ w\ when\ y{=}j] + \boldsymbol{\lambda}}{[total\ words\ when\ y{=}j] + |\mathbf{V}| * \boldsymbol{\lambda}}$$

Parameters to learn

$\tau_{w,1}$	word w
2/15	acting
2/15	and
2/15	amazing
1/15	directing
3/15	great
2/15	movie
2/15	score
1/15	terrible

$\tau_{w,-1}$	word w
1/11	acting
1/11	and
1/11	amazing
2/11	directing
1/11	great
1/11	movie
1/11	score
3/11	terrible

Laplace Smoothing with λ = 1

Laplace Smoothing Avoids Zeroes

- Given test example x^{test} = "great directing"
- Compute p(x, y=+1)
  = p(y=+1) * p(x|y=+1)
  = p(y=+1) * p("great"|y=+1) * p("directing"|y=+1)
  = 3/5 * **3/15 * 1/15** = **0.0080**
- Compute p(x, y=-1)
  = p(y=-1) * p(x|y=-1)
  = p(y=-1) * p("great"|y=-1) * p("directing"|y=-1)
  = 2/5 * **1/11 * 2/11**= **0.0066**
- By Bayes Rule:
  - P(y=+1|x) = 0.0080/(0.0080+0.0066) = **.595**
  - **Model thinks y=+1 is more likely**

Learned Parameters

$\pi_1 = 3/5$ $\pi_{-1} = 2/5$

$\tau_{w,1}$	word w
2/15	acting
2/15	and
2/15	amazing
1/15	**directing**
3/15	**great**
2/15	movie
2/15	score
1/15	terrible

$\tau_{w,-1}$	word w
1/11	acting
1/11	and
1/11	amazing
2/11	**directing**
1/11	**great**
1/11	movie
1/11	score
3/11	terrible

Laplace Smoothing with λ = 1

Problem #2: Numerical Underflow

- Given **long** test example x^{test} = "great directing and acting, amazing score, …"
- Compute $p(x, y=+1)$:
  $= p(y=+1) * p(x|y=+1)$
  $= p(y=+1) * p(\text{"great"}|y=+1) * p(\text{"directing"}|y=+1) * p(\text{"and"}|y=+1) * p(\text{"acting"}|y=+1) * \ldots$
  - If you actually try to do this on a computer, you will get 0!
    - Multiplying many small numbers results in **numerical underflow**
    - Result is so small that it becomes 0

Learned Parameters

$\pi_1 = 3/5$

$\tau_{w,1}$	word w
2/15	acting
2/15	and
2/15	amazing
1/15	directing
3/15	great
2/15	movie
2/15	score
1/15	terrible

$\pi_{-1} = 2/5$

$\tau_{w,-1}$	word w
1/11	acting
1/11	and
1/11	amazing
2/11	directing
1/11	great
1/11	movie
1/11	score
3/11	terrible

Laplace Smoothing with λ = 1

Use Log Space to Avoid Underflow

- Given long test example x^{test} = "great directing and acting, amazing score, …"
- Instead compute ***log* p(x, y=+1)**:

  = *log* $p(y=+1)$ + *log* $p(x|y=+1)$

  = *log* $p(y=+1)$ + *log* $p(\text{"great"}|y=+1)$ + *log* $p(\text{"directing"}|y=+1)$ + *log* $p(\text{"and"}|y=+1)$ + *log* $p(\text{"acting"}|y=+1)$ + …
  - This will not underflow, just adding together some negative numbers
- At test time: compute ***log* p(x, y=j)** for each j, choose the j with largest value

Learned Parameters

$\pi_1 = 3/5$

$\tau_{w,1}$	word w
2/15	acting
2/15	and
2/15	amazing
1/15	directing
3/15	great
2/15	movie
2/15	score
1/15	terrible

$\pi_{-1} = 2/5$

$\tau_{w,-1}$	word w
1/11	acting
1/11	and
1/11	amazing
2/11	directing
1/11	great
1/11	movie
1/11	score
3/11	terrible

Laplace Smoothing with λ = 1

Today's Plan

- Generative vs. Discriminative Classifiers
- Naïve Bayes for Text Classification
  - First Attempt
  - Two fixes to avoid zeroes
- Naïve Bayes for Feature Vectors

Naïve Bayes for Feature Vectors

Text Classification Setting

- Each input *x* is a document
  - Documents can have different numbers of words
  - $x^{(i)}_j$ is *j*-th word of *i*-th training example
  - We made an implicit assumption that **position of words does not matter**—same distribution for 1st word of document, 2nd word, etc.

Feature Vector Setting

- Each input *x* is a feature vector
  - Each vector is of a fixed size d
  - $x^{(i)}_j$ is *j*-th feature of *i*-th training example
  - **Each feature means something different**! Can't treat them the same

Naïve Bayes for Feature Vectors

- Step 1: Each $y^{(i)}$ was sampled from the prior distribution $p(y)$
- Step 2: **For each j = 1, …, d:**
  Feature $x^{(i)}_j$ was sampled independently from the **feature-specific** distribution for label $y^{(i)}$

Task: Predict if user will like album (y) given genre (x_1) and decade (x_2)

Step 1: Choose y

y=1 with probability π_1

y=-1 with probability π_{-1}

Step 2: Sample positive features

Step 2: Sample negative features

$P(x_1 \mid y=1)$	genre
0.31	rock
0.24	pop
0.23	hip hop
…	…

$P(x_2 \mid y=1)$	decade
0.33	2010's
0.28	2020's
0.21	2000's
…	…

$P(x_1 \mid y=-1)$	genre
0.24	country
0.22	rock
0.18	pop
…	…

$P(x_2 \mid y=-1)$	decade
0.35	2020's
0.24	2010's
0.15	1990's
…	…

Most likely x = (rock, 2010's)

Most likely x = (country, 2020's)

Naïve Bayes for Feature Vectors

- How to learn? Count occurrences for each feature
  - E.g., Count how many “liked” albums come from each genre
- Apply Laplace Smoothing to all (label, feature) pairs
  - E.g., Imagine 1 additional album of each genre was liked

Task: Predict if user will like album (y) given genre (x_1) and decade (x_2)

Step 1: Choose y

y=1 with probability π_1

y=-1 with probability π_{-1}

Step 2: Sample positive features

Step 2: Sample negative features

$P(x_1 \mid y=1)$	genre
???	country
???	hip hop
???	pop
…	…

$P(x_2 \mid y=1)$	decade
???	1950’s
???	1960’s
???	1970’s
…	…

$P(x_1 \mid y=-1)$	genre
???	country
???	hip hop
???	pop
…	…

$P(x_2 \mid y=-1)$	decade
???	1950’s
???	1960’s
???	1970’s
…	…

Discriminative vs. Generative Comparison

Logistic/Softmax Regression

- Usually higher accuracy, especially with large dataset
  - P(y|x) usually simpler to learn than P(x|y)
- Can do arbitrary feature processing. Input features can be related to each other, since we don’t make any conditional independence assumptions

Naïve Bayes

- Learning is easier—no gradient descent, just count!
- Can handle missing input features—just ignore them when computing P(x|y)
- Easy to make small updates to the model
  - New training example? Just increment counts
  - New label? Fit P(x|y=new label), everything else stays the same

Summary: Generative Classifiers, Naïve Bayes

- Generative Classifier: Model p(y) and p(x|y)
  - Modeling p(y) is easy (just count how often each label occurs)
  - Modeling p(x|y) is hard
    - Naïve Bayes assumption: Each word/feature of x is **conditionally independent** given y
    - This makes modeling p(x|y) easy: Just count!
    - Need to be careful to avoid zeroes
      - Laplace Smoothing to avoid zero probability of unseen (word, label) pairs
      - Work in log space to avoid numerical underflow
  - Use Bayes Rule to compute prediction p(y|x) from p(y) and p(x|y)