



Computer
Science

CSC380: Principles of Data Science

Predictive Modeling and Classification 3

Xinchen Yu

Midterm score curving

$$new = score + \sqrt{1 - score}$$

origin_scores	new_scores
38	45.87400787401181
39	46.810249675906654
40	47.74596669241483
41	48.681145747868605
42	49.61577310586391
43	50.54983443527075
44	51.48331477354788
45	52.41619848709566
46	53.348469228349536
47	54.28010988928052
48	55.211102550927976
49	56.14142842854285
50	57.071067811865476
51	58.0
52	58.92820323027551
53	59.855654600401046
54	60.782329983125265
55	61.70820393249937
56	62.633249580710796

origin_scores	new_scores
63	69.08276253029823
64	70.0
65	70.91607978309962
66	71.8309518948453
67	72.74456264653803
68	73.65685424949238
69	74.56776436283002
70	75.47722557505166
71	76.3851648071345
72	77.29150262212919
73	78.19615242270663
74	79.09901951359278
75	80.0
76	80.89897948556636
77	81.79583152331271
78	82.69041575982342
79	83.58257569495584
80	84.47213595499957
81	85.35889894354068

origin_scores	new_scores
82	86.24264068711929
83	87.12310562561765
84	88.0
85	88.87298334620742
86	89.74165738677394
87	90.605551275464
88	91.46410161513775
89	92.3166247903554
90	93.16227766016839
91	94.0
92	94.82842712474618
93	95.64575131106459
94	96.44948974278317
95	97.23606797749979
96	98.0
97	98.73205080756888
98	99.41421356237309
99	100.0
100	100.0

Estimating current scores

- Assignments: 36%
- Midterm: 20%
- Project: 14%
- Final Exam: 20%
- Participation: 10%

- 7 homeworks, drop the lowest one.
 - Each 6 points
 - e.g., $6 * 120/130 = 5.54$ -- if a homework total points are 130 and you got 120

Model Evaluation

Suppose our classifier distinguishes between cats and non-cats.
We can make the following table called **confusion matrix**:

Actual class \ Predicted class	Cat	Non-cat
	Cat	Non-cat
Cat	6 true positives	2 false negatives
Non-cat	1 false positive	3 true negatives

It tells us if classifier is biased towards certain mistakes (False Positives, False Neg.)

Good for investigating opportunities to improve the classifier.

		PREDICTED	
		POSITIVE	NEGATIVE
ACTUAL	POSITIVE	TRUE POSITIVES	FALSE NEGATIVES
	NEGATIVE	FALSE POSITIVES	TRUE NEGATIVES

Precision: dividing the true positives by anything that was predicted as a positive.

$$\frac{\text{TRUE POSITIVES}}{\text{TRUE POSITIVES} + \text{FALSE POSITIVES}}$$

		PREDICTED	
		POSITIVE	NEGATIVE
ACTUAL	POSITIVES	TRUE POSITIVES	FALSE NEGATIVES
	NEGATIVE	FALSE POSITIVES	TRUE NEGATIVES

Recall (or True Positive Rate): dividing the true positives by anything that should have been predicted as positive.

$$\frac{\text{TRUE POSITIVES}}{\text{TRUE POSITIVES} + \text{FALSE NEGATIVES}}$$

F1 score symmetrically represents both precision and recall in one metric.




$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}$$

- This is the *harmonic mean* of precision and recall
 - `harmonic_mean(x,y)`

$$\frac{1}{\frac{1}{2}(\frac{1}{x} + \frac{1}{y})}$$

- Gives equal importance to precision and recall – F1 may not be best when you care about one more than the other (e.g., in medical tests we care about recall)

Evaluation functions live in `metrics`

<code>metrics.confusion_matrix(y_true, y_pred, *)</code>	Compute confusion matrix to evaluate the accuracy of a classification.	
<code>metrics.dcg_score(y_true, y_score, *[k, ...])</code>	Compute Discounted Cumulative Gain.	
<code>metrics.det_curve(y_true, y_score[, ...])</code>	Compute error rates for different probability thresholds.	
<code>metrics.f1_score(y_true, y_pred, *[, ...])</code>	Compute the F1 score, also known as balanced F-score or F-measure.	
<code>metrics.fbeta_score(y_true, y_pred, *, beta)</code>	Compute the F-beta score.	
<code>metrics.hamming_loss(y_true, y_pred, *[, ...])</code>	Compute the average Hamming loss.	
<code>metrics.hinge_loss(y_true, pred_decision, *)</code>	Average hinge loss (non-regularized).	
<code>metrics.jaccard_score(y_true, y_pred, *[, ...])</code>	Jaccard similarity coefficient score.	
<code>metrics.log_loss(y_true, y_pred, *[, eps, ...])</code>	Log loss, aka logistic loss or cross-entropy loss.	
<code>metrics.matthews_corrcoef(y_true, y_pred, *)</code>	Compute the Matthews correlation coefficient (MCC).	
<code>metrics.multilabel_confusion_matrix(y_true, ...)</code>	Compute a confusion matrix for each class or sample.	
<code>metrics.ndcg_score(y_true, y_score, *[k, ...])</code>	Compute Normalized Discounted Cumulative Gain.	
<code>metrics.precision_recall_curve(y_true, ...)</code>	Compute precision-recall pairs for different probability thresholds.	
<code>metrics.precision_recall_fscore_support(...)</code>	Compute precision, recall, F-measure and support for each class.	
<code>metrics.precision_score(y_true, y_pred, *[, ...])</code>	Compute the precision.	
<code>metrics.recall_score(y_true, y_pred, *[, ...])</code>	Compute the recall.	

Naïve Bayes

Heads Up This section will return to some math as we go in depth. But, much of it is that you already know with a new application (Naïve Bayes Classification) – **ask questions if you are lost**

- Introduction to Naïve Bayes Classifier
- Maximum Likelihood Estimation

Math Prep

- N RVs conditionally independent, given Z, if and only if:

$$p(X_1, \dots, X_N \mid Z) = \prod_{i=1}^N p(X_i \mid Z)$$

Probability 3, page 25

- Bayes' rule

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

Probability 2, page 24

- Maximum likelihood estimation
- Bernoulli and Gaussian distribution

Statistics 2

Probability 3, 4
Statistics 2

Intuition

Training Data:

Person	height (feet)
male	6
male	5.92 (5'11")
male	5.58 (5'7")
male	5.92 (5'11")
female	5
female	5.5 (5'6")
female	5.42 (5'5")
female	5.75 (5'9")

↑
Y

↑
X

Task: observe feature x_n , predict label $y_n \in (0, 1)$

Choose the class with higher probability:

$$P(Y_n = 1|X_n = x_n) \text{ vs } P(Y_n = 0|X_n = x_n)$$

$$P(Y_n = 1|X_n = x_n) = \frac{P(Y_n = 1, X_n = x_n)}{P(X_n = x_n)}$$

$$P(Y_n = 0|X_n = x_n) = \frac{P(Y_n = 0, X_n = x_n)}{P(X_n = x_n)}$$

How to learn a joint probability model for $P(Y, X)$?

Training Data:

Person	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9



Task: Observe features x_1, \dots, x_D and predict class label $y \in \{1, \dots, C\}$

Model: Assume that the feature x and its label y follows certain type of distribution \mathcal{D} with parameter θ .

$$(x, y) \stackrel{\text{i.i.d.}}{\sim} \mathcal{D}_\theta$$

Training Algorithm: Estimate θ e.g., MLE $\hat{\theta}$

To classify: Compute

$$\hat{y} = \arg \max_{c \in \{1, \dots, C\}} p(y = c \mid x; \hat{\theta})$$

what comes after semicolon is the parameter of the distribution

Training Data:

Person	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9



Features

Task: Observe features x_1, \dots, x_D and predict class label $y \in \{1, \dots, C\}$

Naïve Bayes Model: Treat features as *conditionally independent* given class label,

$$p(x, y) = p(y)p(x|y) = p(y) \prod_{d=1}^D p(x_d | y)$$

build individual models for these

To classify a given instance x : Bayes rule!

$$p(y = c | x) = \frac{p(y = c)p(x | y = c)}{p(x)}$$

Key concept in Naïve Bayes

$$p(x, y) = p(y) p(x|y) = p(y) \prod_{d=1}^D p(x_d | y)$$

Class prior distribution

Class conditional distribution

Given one data point, it has 4 features (input), and the label is 0 (output)

$$\begin{aligned} p(x_1, x_2, x_3, x_4, y = 0) &= p(y = 0) \cdot p(x_1, x_2, x_3, x_4 | y = 0) \\ &= p(y = 0) \cdot p(x_1 | y = 0) \cdot p(x_2 | y = 0) \cdot p(x_3 | y = 0) \cdot p(x_4 | y = 0) \end{aligned}$$

$$p(x, y) = p(y) \underbrace{p(x|y)}_{\text{Class prior distribution}} = p(y) \prod_{d=1}^D \underbrace{p(x_d | y)}_{\text{Class conditional distribution}}$$

Class prior distribution

Class conditional distribution

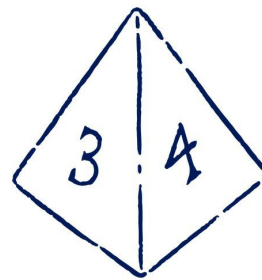
For the **class prior distribution**, take categorical distribution.

$$y \sim \text{Categorical}(\pi), \quad \pi \in \mathbb{R}^C, \pi_c \geq 0, \sum_c \pi_c = 1$$

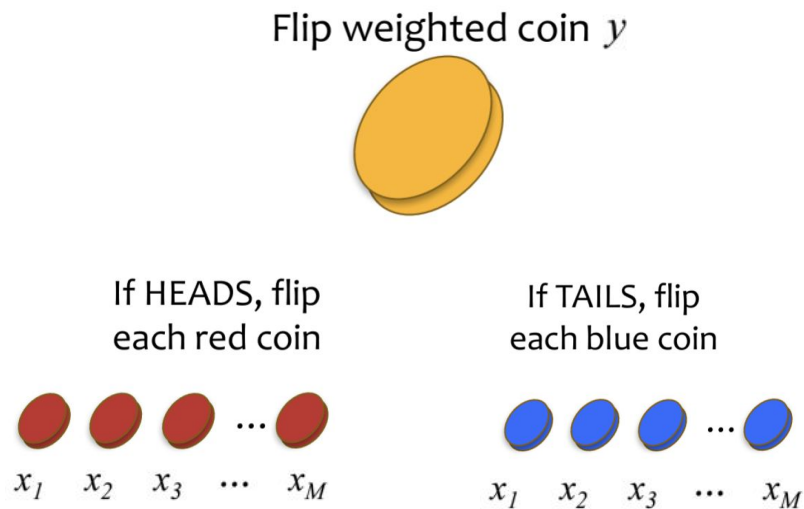
$$\Rightarrow p(y = c) = \pi_c$$

Example: biased 4-sided die Y, given:

- $P(Y=1) = 0.2, \quad P(Y=2) = 0.3, \quad P(Y=3) = 0.1, \quad P(Y=4) = 0.4$



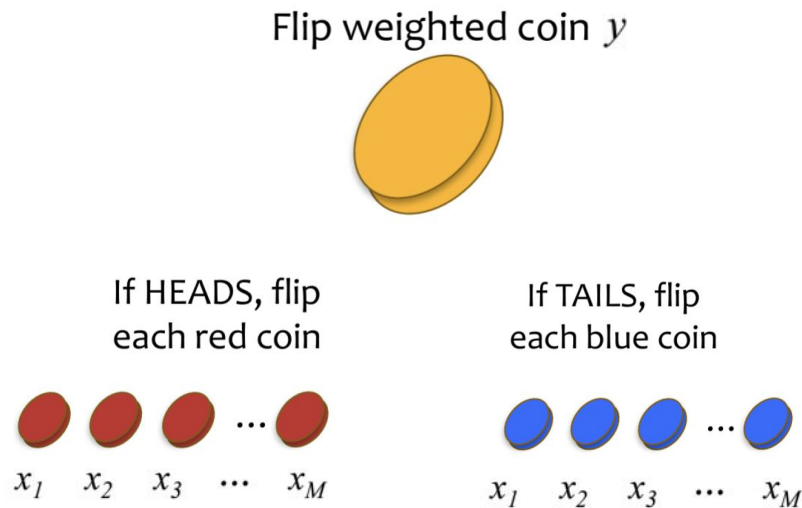
Class prior distribution



y	x_1	x_2	x_3	...	x_M
0	1	0	1	...	1
1	0	1	0	...	1
1	1	1	1	...	1
0	0	0	1	...	1
0	1	0	1	...	0
1	1	0	1	...	0

$p(y)$

Class conditional distribution

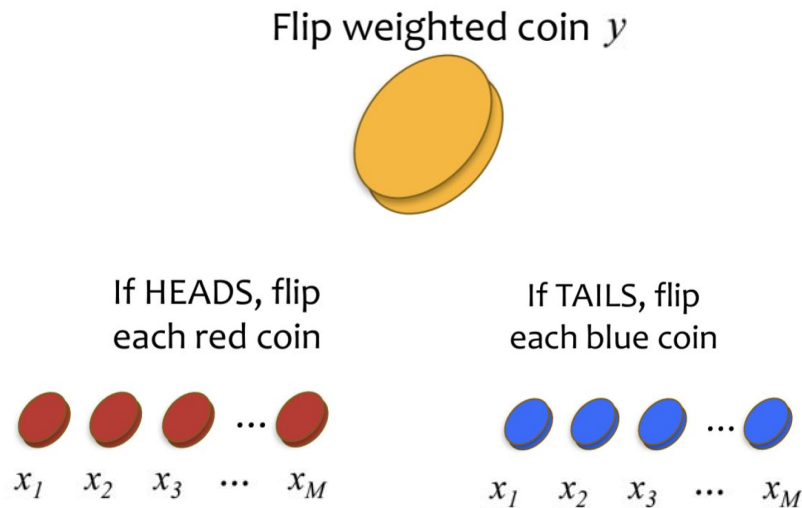


Each red / blue coin biases can be different.

y	x_1	x_2	x_3	...	x_M
0	1	0	1	...	1
1	0	1	0	...	1
1	1	1	1	...	1
0	0	0	1	...	1
0	1	0	1	...	0
1	1	0	1	...	0

$p(x_1|y=1)$

Class conditional distribution

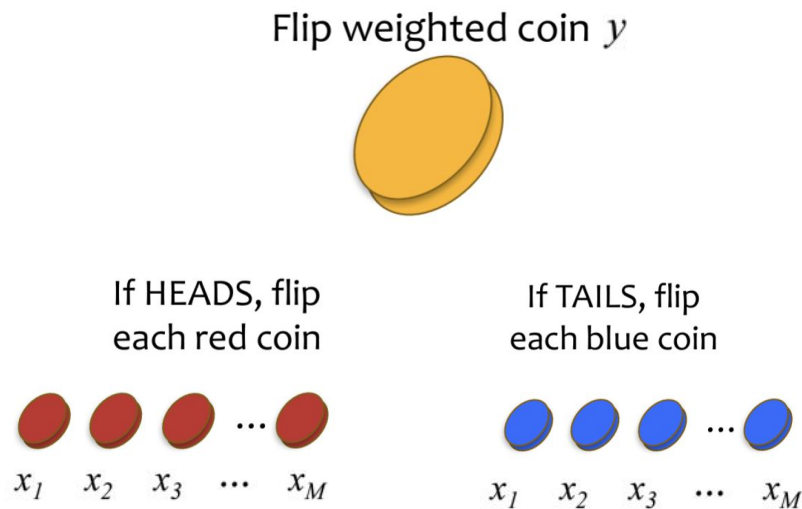


Each red / blue coin biases can be different.

y	x_1	x_2	x_3	...	x_M
0	1	0	1	...	1
1	0	1	0	...	1
1	1	1	1	...	1
0	0	0	1	...	1
0	1	0	1	...	0
1	1	0	1	...	0

$p(x_2|y = 1)$

Class conditional distribution



Each red / blue coin biases can be different.

y	x_1	x_2	x_3	...	x_M
0	1	0	1	...	1
1	0	1	0	...	1
1	1	1	1	...	1
0	0	0	1	...	1
0	1	0	1	...	0
1	1	0	1	...	0

\uparrow
 $p(x_1|y=0)$

Simplifying Assumption: “*Class conditional*” distribution assumes features are conditionally independent given class

$$p(x | y) = \prod_{d=1}^D p(x_d | y)$$

- “Naïve” as in general features are likely to be dependent.
- Every feature can have a different class-conditional distribution



$P(x_1|y)$ can be different from $P(x_2|y)$

Features are typically not independent!

Example 1 If a recent news article contains word “Donald” it is much more likely to contain the word “Trump”.

Example 2 If flower petal width is very large then petal length is also likely to be high.



Simplifying Assumption: “*Class conditional*” distribution assumes features are conditionally independent given class

$$p(x | y) = \prod_{d=1}^D p(x_d | y)$$

- “Naïve” as in general features are likely to be dependent.
- Every feature can have a different class-conditional distribution

Doesn't capture correlation among features. But why would it be a good idea?

- Easy computation: For C classes and D features only $O(CD)$ parameters
- Prevents overfitting
- Simplicity

For real-valued features we can use Normal distribution:

$$p(x | y = c) = \prod_{d=1}^D \mathcal{N}(x_d | \mu_{cd}, \sigma_{cd}^2)$$

quiz candidate

Q: how many parameters?

2cd

Parameters of featured for class c

For binary features $x_d \in \{0,1\}$ can use Bernoulli distributions:

$$p(x | y = c) = \prod_{d=1}^D \text{Bernoulli}(x_d | \theta_{cd})$$

quiz candidate

Q: how many parameters?

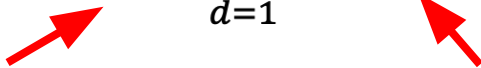
cd

“Coin bias” for d^{th} feature
and class c

- K-valued discrete features: use Categorical.
- Can mix-and-match, e.g. some discrete, some continuous features

$$p(x | y = c) = \prod_{d=1}^{D'} \text{Bernoulli}(x_d | \theta_{cd}) \prod_{d=D'+1}^D \mathcal{N}(x_d | \mu_{cd}, \sigma_{cd}^2)$$

Fitting the model requires learning all parameters...

$$p(x, y = c) = p(y = c; \pi) \prod_{d=1}^D p(x_d | \theta_{cd})$$


Class Prior Parameters

Likelihood Parameters

Given training data $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^N$ maximize the likelihood function,

$$\theta^{\text{MLE}} = \arg \max_{\pi, \theta} \log p(\mathcal{D}; \pi, \theta)$$

$$\theta^{\text{MLE}} = \arg \max_{\pi, \theta} \log p(\mathcal{D}; \pi, \theta) \quad (\mathcal{D} := \{(x^{(i)}, y^{(i)})\}_{i=1}^m)$$

Since data are iid

$$= \arg \max_{\pi, \theta} \log \prod_{i=1}^m p(x^{(i)}, y^{(i)}; \pi, \theta)$$

$\log(ab) = \log a + \log b$

$$= \arg \max_{\pi, \theta} \sum_{i=1}^m \log p(x^{(i)}, y^{(i)}; \pi, \theta)$$

Conditional probability +
Naïve Bayes assumption

$$= \arg \max_{\pi, \theta} \sum_{i=1}^m \log p(y^{(i)}; \pi) + \sum_{i=1}^m \sum_{d=1}^D \log p(x_d^{(i)} | y^{(i)}; \theta_{y^{(i)}d})$$

θ_{cd} : parameter for feature d for class c

Find zero-gradient if concave, or gradient-based optimization otherwise

Analogy:

y	x_1	x_2	...	x_D		
5	0	1	1	0	1	0
2	1	0	0	0	0	0
3	1	1	1	1	0	0
...	...					
4	0	0	1	1	0	1

Let each feature follow a Bernoulli distribution then the model is...

$$y \sim \text{Categorical}(\pi) \quad x_d \mid y = c \sim \text{Bernoulli}(\theta_{cd})$$

The Naïve Bayes joint distribution is then:

$$\begin{aligned} p(\mathcal{D}; \pi, \theta) &= \prod_{i=1}^m \left(p(y^{(i)}; \pi) \prod_d p(x_d^{(i)}; \theta_{y^{(i)}d}) \right) \\ &= \prod_{i=1}^m \left(\prod_c \left(\pi_c^{\mathbf{I}\{y_i=c\}} \prod_j p(x_{ij} | \theta_{jc})^{\mathbf{I}\{y_i=c\}} \right) \right) \end{aligned}$$

Write down log-likelihood and optimize...

j : feature, c : label, i : data

$$y \sim \text{Categorical}(\pi_c) : p(y = c) = \pi_c$$

$$p(y = 1) = \pi_1$$

$$p(y = 2) = \pi_2$$

$$p(y = 3) = \pi_3 = 1 - \pi_1 - \pi_2$$

Q: how many parameters?

$c-1+cj$

$$x|y \sim \text{Bernoulli}(\theta_{jc}) : p(x|y) = \theta_{jc}^x (1 - \theta_{jc})^{1-x}$$

$$x_{j=1}|y = 1 \sim \text{Bernoulli}(\theta_{j=1,c=1}) \quad x_{j=2}|y = 1 \sim \text{Bernoulli}(\theta_{j=2,c=1})$$

$$x_{j=1}|y = 2 \sim \text{Bernoulli}(\theta_{j=1,c=2}) \quad x_{j=2}|y = 2 \sim \text{Bernoulli}(\theta_{j=2,c=2})$$

$$x_{j=1}|y = 3 \sim \text{Bernoulli}(\theta_{j=1,c=3}) \quad x_{j=2}|y = 3 \sim \text{Bernoulli}(\theta_{j=2,c=3})$$

$$\prod_{i=1}^5 \left(\left(\pi_{c=1}^{I(y_i=1)} \cdot p(x_{i,j=1}|\theta_{j=1,c=1}) \cdot p(x_{i,j=2}|\theta_{j=2,c=1}) \right) \cdot \left(\pi_{c=2}^{I(y_i=2)} \cdot p(x_{i,j=1}|\theta_{j=1,c=2}) \cdot p(x_{i,j=2}|\theta_{j=2,c=2}) \right) \cdot \left(\pi_{c=3}^{I(y_i=3)} \cdot p(x_{i,j=1}|\theta_{j=1,c=3}) \cdot p(x_{i,j=2}|\theta_{j=2,c=3}) \right) \right)$$

y	x_1	x_2
1	0	1
3	1	0
3	1	1
2	0	0
1	1	0

Let $m_c := \sum_{i=1}^m \mathbf{I}\{y^{(i)} = c\}$ be number of training examples in class c then,

$$\sum_{i=1}^m \log p(\mathcal{D}; \pi, \theta) = \sum_{c=1}^C m_c \log \pi_c + \sum_{c=1}^C \sum_{i: y^{(i)}=c} \sum_{d=1}^D \log p(x_d^{(i)}; \theta_{cd})$$

Log-likelihood function is concave in all parameters so...

1. Take derivatives with respect to π and θ separately.
2. Set derivatives to zero and solve

$$\hat{\pi}_c = \frac{m_c}{m}$$

**Fraction of training
examples from class c**

$$\hat{\theta}_{cd} = \frac{m_{cd}}{m_c}$$

**Number of “heads” in
training set from class c**

$$m_{cd} = \sum_{i=1}^m \mathbf{I}\{y^{(i)} = c, x_d^{(i)} = 1\}$$

Analogy:

y	x_1	x_2	...	x_D		
5	0	1	1	0	1	0
3	1	0	0	0	0	0
3	1	1	1	1	0	0
...
4	0	0	1	1	0	1

$$\hat{\pi}_c = \frac{m_c}{m}$$

$$\hat{\theta}_{cd} = \frac{m_{cd}}{m_c}$$

$$m_{cd} = \sum_{i=1}^m I\{y^{(i)} = c, x_d^{(i)} = 1\}$$

$$\hat{\pi}_3 = \frac{\# \text{ number of } y = 3}{\# \text{ total data}}$$

$$\hat{\theta}_{3,2} = \frac{\# \text{ number of } y = 3 \text{ and } x_2 = 1}{\# \text{ number of } y = 3}$$

Bernoulli Naïve Bayes: making prediction

$$\hat{\pi}_c = \frac{m_c}{m}$$

$$\hat{\theta}_{cd} = \frac{m_{cd}}{m_c}$$

Given one data point, it has 4 features (input), compare the probabilities:

$$\begin{aligned} p(x_1, x_2, x_3, x_4, y = 0) &= p(y = 0) \cdot p(x_1, x_2, x_3, x_4 | y = 0) \\ &= p(y = 0) \cdot p(x_1 | y = 0) \cdot p(x_2 | y = 0) \cdot p(x_3 | y = 0) \cdot p(x_4 | y = 0) \end{aligned}$$

$$\begin{aligned} p(x_1, x_2, x_3, x_4, y = 1) &= p(y = 1) \cdot p(x_1, x_2, x_3, x_4 | y = 1) \\ &= p(y = 1) \cdot p(x_1 | y = 1) \cdot p(x_2 | y = 1) \cdot p(x_3 | y = 1) \cdot p(x_4 | y = 1) \end{aligned}$$

Bernoulli Naïve Bayes MLE: issue

no data points of class 2:

$$p(y = 2) = 0$$

no data points in class 1 & 3 is 1:

$$p(x_1 = 0|y = 3) = 0$$

$$p(x_1 = 0|y = 1) = 0$$

y	x_1	x_2
1	0	1
3	0	0
3	0	1
3	0	0
1	0	0
?	1	0

What if there are *no* examples of class c in the training set?

$$\hat{\pi}_c = 0 \quad \text{Model will never learn to guess class } c$$

What if all data points i in class c has $x_d^{(i)} = 0$ in the training set?

$$\hat{\theta}_{cd} = 0$$

Model will assign 0 likelihood for test data with $x_d = 1$ for class c (i.e., $p(x|y = c)$).

What does it imply on $p(y = c|x)$? 0!

Training data needs to see every possible outcome for each feature

Any ideas how we can fix this problem?

We could add a small constant to prevent zero probabilities...

$$\hat{\pi}_c \propto m_c + \alpha$$

$$\hat{\theta}_{cj} \propto m_{cj} + \beta$$

$$\alpha, \beta > 0$$

**Pseudocounts
add- α Smoothing
Laplace smoothing**

....

typical choice: set $\alpha = \beta = 1$

Another smoothing method:

$$\hat{P}(w_i|c) = \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)} = \frac{\text{count}(w_i, c) + 1}{(\sum_{w \in V} \text{count}(w, c)) + |V|}$$

Word count in category c

**Vocabulary size
in whole corpus**

Naïve Bayes in Sentiment Classification

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

$$\hat{\pi}_c = \frac{m_c}{m} \quad P(-) = \frac{3}{5} \quad P(+) = \frac{2}{5}$$

Naïve Bayes in Sentiment Classification

$$\hat{P}(w_i|c) = \frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)}$$

smoothing ↓

$$\frac{\text{count}(w_i, c) + 1}{(\sum_{w \in V} \text{count}(w, c)) + |V|}$$

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Vocabulary size

$$20 = 14 + 9 - 3$$

3: the, and, very (duplicate)

$$P(\text{"predictable"}|-) = \frac{1+1}{14+20}$$

$$P(\text{"predictable"}|+) = \frac{0+1}{9+20}$$

$$P(\text{"no"}|-) = \frac{1+1}{14+20}$$

$$P(\text{"no"}|+) = \frac{0+1}{9+20}$$

$$P(\text{"fun"}|-) = \frac{0+1}{14+20}$$

$$P(\text{"fun"}|+) = \frac{1+1}{9+20}$$

Naïve Bayes in Sentiment Classification

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Ignore unknown words



$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

The model thus predicts the class *negative* for the test sentence.

Scikit-learn has separate classes each feature type

`sklearn.naive_bayes.GaussianNB`

Real-valued features

`sklearn.naive_bayes.MultinomialNB`

Discrete K-valued feature counts (e.g. multiple die rolls)

`sklearn.naive_bayes.BernoulliNB`

Binary features (e.g. coinflip)

`sklearn.naive_bayes.CategoricalNB`

Discrete K-valued features (e.g. single die roll)

For large training data that don't fit in memory use Scikit-learn's out-of-core learning