Le
Learning

# Machine Learning: Naïve Bayes Classifiers

CSE 415: Introduction to Artificial Intelligence
University of Washington
Winter, 2018





Le
Learning

## Outline

- Motivation
- A discrete example: Classifying fruits
- The Naïve Bayes assumption
- Maximum likelihood estimation of probabilities from samples
- A continuous example using a Gaussian model: Classifying online shoppers.



Le
Learning

## Motivation

•Bayes' rule is a general technique in classification, but costly in terms of requiring large training sets.
•By making independence assumptions, much less training data is required.
•Often the results are very good.
•Naïve Bayes classifiers are based on the assumption that likelihoods of each feature are independent of those of other features.



Le
Learning

## A Discrete Example

**A training example consists of a vector of attribute values with a category indicator.**

Attribute values
Category indicators

⟨ ⟨ long, yellow ⟩, *banana* ⟩
⟨ ⟨ not long, yellow⟩, *lemon* ⟩
⟨ ⟨ long, not yellow⟩, *other*⟩
⟨ shape, color, ⟩

Attribute names

# Example Training Data Stats

| Long | Yellow | class | count |
|---|---|---|---|
| No | No | Lemon | 2 |
| No | No | Banana | 0 |
| No | No | Other | 3 |
| No | Yes | Lemon | 5 |
| No | Yes | Banana | 0 |
| No | Yes | Other | 1 |
| Yes | No | Lemon | 0 |
| Yes | No | Banana | 3 |
| Yes | No | Other | 2 |
| Yes | Yes | Lemon | 0 |
| Yes | Yes | Banana | 9 |
| Yes | Yes | Other | 0 |



# The Training Data Stats (cont.)

| Long | Yellow | class | count |
|---|---|---|---|
| No | No | Lemon | 2 |
| No | No | Banana | 0 |
| No | No | Other | 3 |
| No | Yes | Lemon | 5 |
| No | Yes | Banana | 0 |
| No | Yes | Other | 1 |
| Yes | No | Lemon | 0 |
| Yes | No | Banana | 3 |
| Yes | No | Other | 2 |
| Yes | Yes | Lemon | 0 |
| Yes | Yes | Banana | 9 |
| Yes | Yes | Other | 0 |

Compute the priors:
P(lemon) = 7 / 25
P(banana) = 12 / 25
P(other) = 6 / 25

P(long) = 14 / 25
P(yellow) = 15 / 25

Compute the likelihoods of individual features:
P(long | lemon) = 0
P(long | banana) = 12/12
P(long | other) = 2/6

P(yellow | lemon) = 5/7
P(yellow | banana) = 9/12
P(yellow | other) = 1/6



# To Classify An Instance

⟨ ⟨**not long, yellow** ⟩, **?** ⟩
Let's call the vector ⟨not long, yellow⟩ the *evidence* E.
Ideally, we would get the *a posteriori* probability of each class and choose the class with the highest:
P(lemon | E), P(banana | E), P(other | E).
This would require applying Bayes' rule as follows, e.g., for lemon:

P(lemon | E) = P( E | lemon) P(lemon) / P(E)
However, we don't have P(E | lemon) in the given feature likelihoods we calculated.



# Likelihood Computation

We could, in principle, compute P(E | lemon) using this:
P(not long, yellow | lemon) =
P(not long | lemon and yellow) P(yellow | lemon).

But we don't have the first of these readily available, either.

It's a lot easier if we assume that P(long | lemon) and P(yellow | lemon) are independent.

Then we can approximate P(E | lemon) as
P(not long and yellow | lemon ) ≈
(1- P(long | lemon) ) P(yellow | lemon)

# Classification with N.B.

P(lemon | E) = P(E | lemon) P(lemon) / P(E)
≈ (1- P(long | lemon) ) P(yellow | lemon) P(lemon) / P(E)
= (1 - 0) ( 5/7 ) (7/25) / P(E)
= (35/185) / P(E) = **(1/5) / P(E)**
Similarly,
P(banana | E) ≈
(1 - P(long | banana)) P(yellow | banana) P(banana) / P(E)
= (1 - 12/12) (9/12) (12/25) / P(E)
= **0**
and P(other | E) ≈
(1 - long| other)) P(yellow | other) P( other) / P(E)
= (1 - 2/6) (1/6) (6/25) / P(E)
= (2/3)(1/25) / P(E) = **(2/75) / P(E)**
Clearly P(lemon | E) gets a higher value than P(other | E),
and so the new instance is classified as a lemon (assuming we are using the Maximum A Posteriori (MAP) probability classification rule).



# Exercise

Using the same Naïve Bayes classifier,

Classify:
⟨⟨ not long, not yellow ⟩, ? ⟩

1. What is P(lemon | E) ?
2. What is P(banana | E) ?
3. What is P(other | E) ?

What is $\text{argmax}_f$ P(f | E) ?

priors:
P(lemon) = 7 / 25
P(banana) = 12 / 25
P(other) = 6 / 25

P(long) = 14 / 25
P(yellow) = 15 / 25

likelihoods of individual features:
P(long | lemon) = 0
P(long | banana) = 12/12
P(long | other) = 2/6

P(yellow | lemon) = 5/7
P(yellow | banana) = 9/12
P(yellow | other) = 1/6



# A Difficulty When Using Frequencies for Probabilities

| Long | Yellow | class | count |
|---|---|---|---|
| No | No | Lemon | 2 |
| No | No | Banana | 0 |
| No | No | Other | 3 |
| No | Yes | Lemon | 5 |
| No | Yes | Banana | 0 |
| No | Yes | Other | 1 |
| Yes | No | Lemon | 0 |
| Yes | No | Banana | 3 |
| Yes | No | Other | 2 |
| Yes | Yes | Lemon | 0 |
| Yes | Yes | Banana | 9 |
| Yes | Yes | Other | 0 |

Compute the priors:
P(lemon) = 7 / 25
P(banana) = 12 / 25
P(other) = 6 / 25

P(long) = 14 / 25
P(yellow) = 15 / 25

Compute the likelihoods of individual features:
P(long | lemon) = 0
P(long | banana) = 12/12
P(long | other) = 2/6

P(yellow | lemon) = 5/7
P(yellow | banana) = 9/12
P(yellow | other) = 1/6

Small training sets tend to lead to some zeros in the counts, even when the underlying distributions have nonzero probabilities.
These zeros can wreak havoc with Naive Bayes classifiers.
Therefore ....



# Likelihoods from Training Data are Typically Inadequate

| Long | Yellow | class | count |
|---|---|---|---|
| No | No | Lemon | 2 |
| No | No | Banana | 0 |
| No | No | Other | 3 |
| No | Yes | Lemon | 5 |
| No | Yes | Banana | 0 |
| No | Yes | Other | 1 |
| Yes | No | Lemon | 0 |
| Yes | No | Banana | 3 |
| Yes | No | Other | 2 |
| Yes | Yes | Lemon | 0 |
| Yes | Yes | Banana | 9 |
| Yes | Yes | Other | 0 |

If P(long, yellow, other)=0, this is saying that it is impossible to have a long, yellow fruit other than banana or lemon!

It also implies that
P(other | long, yellow) = 0

What if we had no training data? Using our default approach, we'd have to assume that everything is impossible.

It is usually more reasonable to assume that everything is possible.

# A Long, Yellow Papaya



# Using Maximum Likelihood to Estimate the Probability Distribution

Instead of taking the count ratios as probabilities, use them as evidence for underlying distributions and estimate those distributions.

A set of distributions can be parameterized by $\theta$. So choose the best $\theta$: the one that has the highest likelihood given the training data.

$$\theta_{best} \leftarrow \text{argmax}_{\theta} \; P(\theta \mid T)$$

We can estimate the parameters of $\theta$ by including phantom examples in our training set to make sure that no count is 0. By adding 1 to every count, we get frequency ratios that generally do not add up to 1 any more, and so we normalize the new ratios so they DO add up to 1.



# Laplace Smoothing

| Long | Yellow | class | count |
|---|---|---|---|
| No | No | Lemon | 2 + 1 |
| No | No | Banana | 0 + 1 |
| No | No | Other | 3 + 1 |
| No | Yes | Lemon | 5 + 1 |
| No | Yes | Banana | 0 + 1 |
| No | Yes | Other | 1 + 1 |
| Yes | No | Lemon | 0 + 1 |
| Yes | No | Banana | 3 + 1 |
| Yes | No | Other | 2 + 1 |
| Yes | Yes | Lemon | 0 + 1 |
| Yes | Yes | Banana | 9 + 1 |
| Yes | Yes | Other | 0 + 1 |

For each combination of features, add k phantom training examples.

Typically, use k=1.

This means:
If no training data, we assume a uniform distribution.

Especially using Naive Bayes classification, Laplace smoothing helps avoid overfitting.

Overfitting: A learned classifier performs well enough on the training examples, but performs poorly on the test data.



# Naïve Bayes Classifiers with Continuous-Valued Features

Let $X = \langle x_0, x_1, \ldots, x_{n-1} \rangle$ be a vector in $R^n$.

We can still use Bayes' rule to compute posterior values useful for classification.

However, the values will be probability density values rather than probabilities.
$P(y \mid X) = P(X \mid y)P(y) \,/\, P(X)$

The Naïve Bayes assumption of conditional independence is
$P(X \mid y) = P(x_0 \mid y)\, P(x_1 \mid y) \ldots P(x_{n-1} \mid y)$

## Continuous Features with Gaussian Distributions

Let's assume each $x_i$ in $\langle x_0, x_1, \ldots, x_{n-1} \rangle$ comes from a probability distribution given by the probability density function (pdf): $P(x_i = x \mid y=c) = \exp((x - \mu_{i,c})^2 / 4\sigma_{i,c}^2)$

The Naïve Bayes assumption is
$P(X \mid y=c) = \exp((x - \mu_{0,c})^2 / 4\sigma_{0,c}^2) \ldots \exp((x - \mu_{n-1,c})^2 / 4\sigma_{n-1,c}^2)$

Take the logarithm of both sides:
$\mathbf{ln}\, P(X \mid y=c) = (x - \mu_{0,c})^2 / 4\sigma_{0,c}^2 + \ldots + (x - \mu_{n-1,c})^2 / 4\sigma_{n-1,c}^2$

To classify example X, find $\text{argmax}_c\, P(X \mid y=c)\, P(y=c)$.



## Example of N.B. with Gaussians

Consider the case of the Northwest Hiking Supply Company, trying to increase their online sales. When a potential customer visits their website, they want to be able to predict, after 1 minute, whether that visitor is a hot prospect, so that they can offer a special "closer" promotion. For training purposes, any past visitor who has purchased something during the visit is considered "hot."

By the end of 60 seconds at the website, they have two feature values: "dwell time" (D) and "number of product revisits" (R). The range of values of D is 0 to 60, and for R it is 0 to 10. From these two measures, they want to classify the visitor as "hot" (H) or "not" (N).



## Training Data for NW Hiking Example

Total visitors: 200
Hot prospects: 5
Not hot: 195

Let's assume we have obtained the means and standard deviations of each of the two subpopulations by the usual methods for getting the parameters of a normal distribution from samples.

Mean dwell time for hot prospects: 38
Mean dwell time for not: 13

Standard deviation for hot prospects: 10
Standard deviation for not: 7

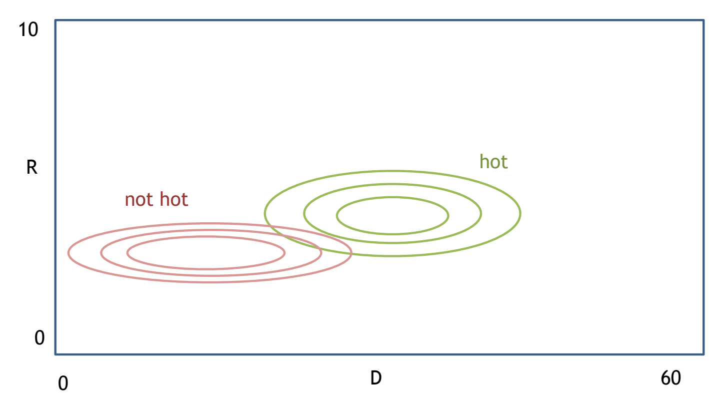
Mean repeat count for hot prospects: 6
Mean repeat count for not: 2

Standard deviation for hot prospects: 3
Standard deviation for not: 2

We'll assume that features D and R are conditionally independent.



## Graphical Display for 2-Feature Example.



Le
Learning

# Conclusions

Naïve Bayes Classifiers require only a relatively small amount of training examples (linear in the number of features times number of values, in the discrete case).

Whereas the full joint distribution typically requires $\Omega(2^n)$ training examples, which is intractable when n is large (e.g., $n > 50$).

Naïve Bayes classification is fast.

The Naïve Bayes assumption of conditional independence, while not usually an accurate model of the underlying joint distribution, still works remarkably well.