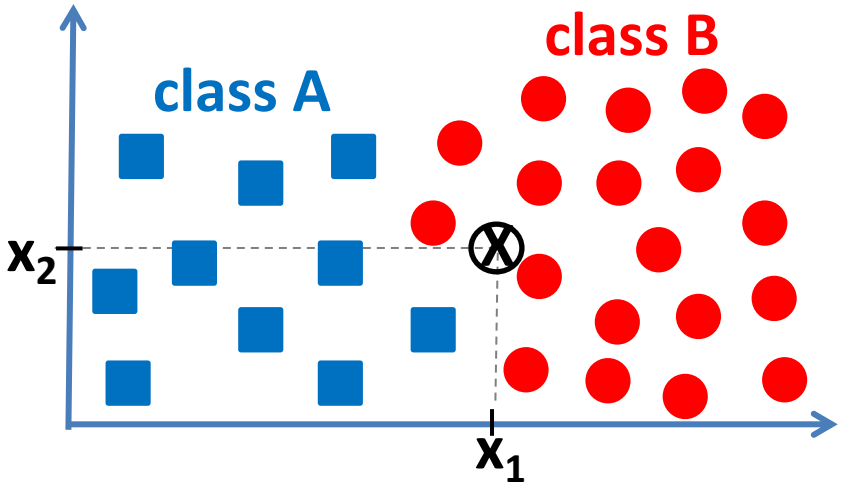
Machine Learning

Lecture 07

Naïve-Bayes model

Intuition on NB

- Which class does **x** belong to?



Class (prior) probabilities tell us how dominant each individual class is over the whole dataset

p(classA) = 10/30 = 33.3%

p(classB) = 20/30 = 66.7%

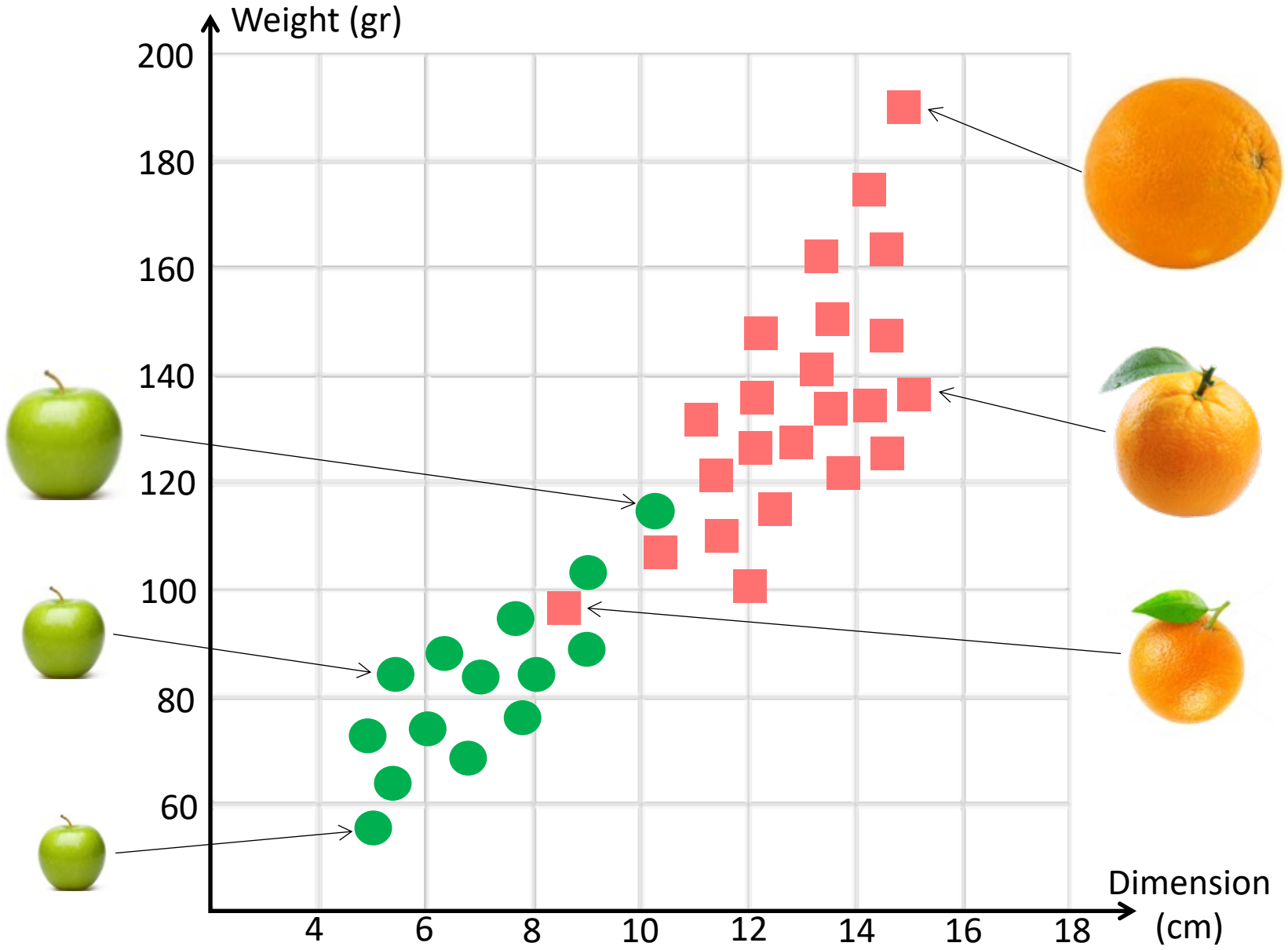
- By examining prior probabilities, we can conclude:

```
if P(classB) > P(classA)
        then classB
else classA
```

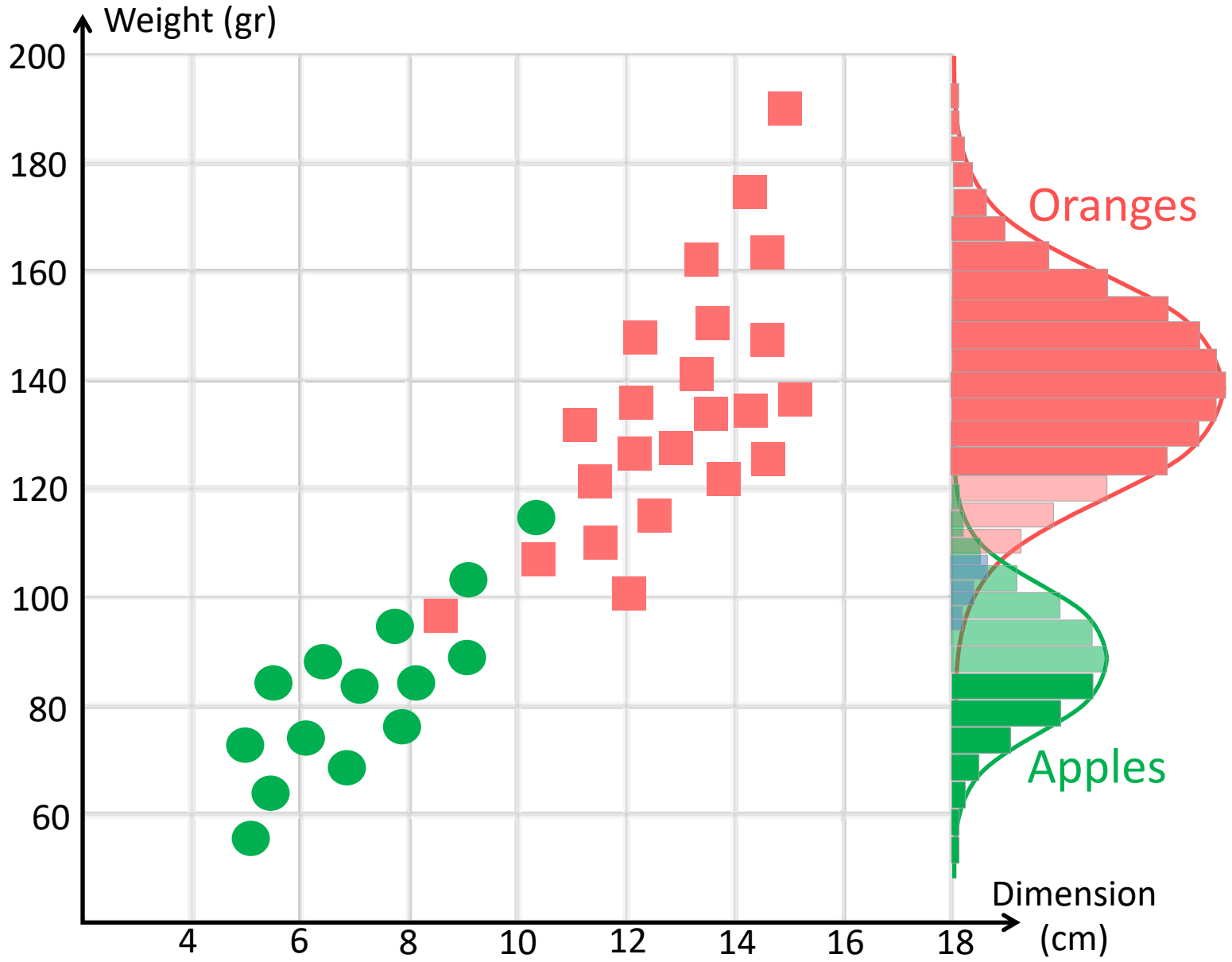
These rules don't tell the whole story though. What we really need is a comparison between conditional probabilities. Given the point X=(x_1, x_2), what is the probability that X belongs to A (or B)?

`p(classB|X) > p(classA|X)?`

Intuition on NB



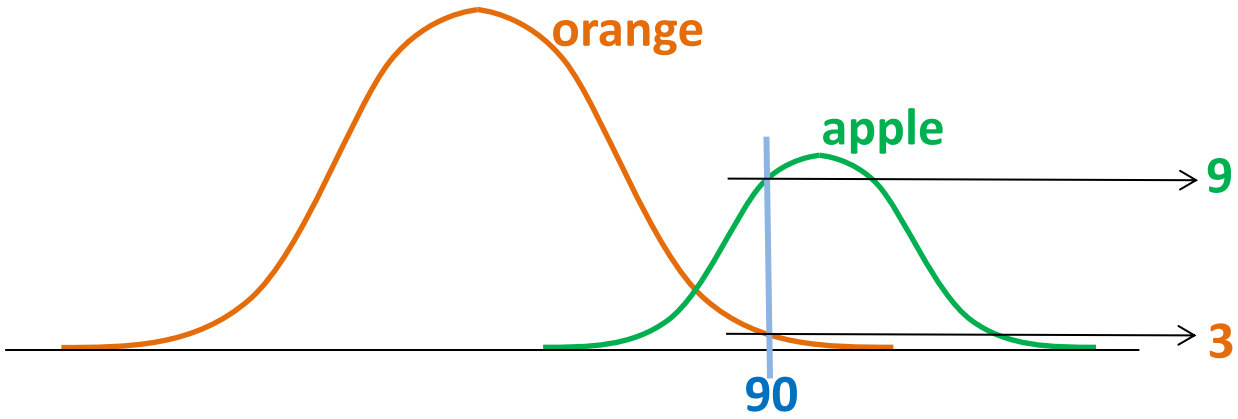
Intuition on NB



Intuition on NB

- We want to classify a fruit based on its weight (90 grams)?
- Given the probability distributions of weight for oranges and apples, we can find out which class is more likely?

  p(class|w=90)=probability of a class given an observed w



  p(**orange**|weight=90)=**3**/(**9**+**3**) = **0.25**

  p(**apple**|weight=90)=**9**/(**9**+**3**) = **0.75**

- For classifications based on dimension and weight, you need to construct a joint probability distribution based on weights and dimensions.

Naive-Bayes

- Naive-Bayes algorithm for predictive modeling
  - A statistical classifier that can predict class membership probabilities (a generative method)
  - Instead of mapping the instance to a class, assign the class with the maximum probability given the instance
  - An eager learner (ideal for real-time predictions)
  - Despite its simplicity, it's surprisingly accurate
  - Somewhat immune to overfitting
    - Suffers from high bias and exhibits low variance
  - Underlying probabilistic model: The Bayes Theorem
  - Basic assumption: Given a class, every attribute is independent of all other attributes (the value of one feature has no effect on that of another – thus the word "Naive"... more on this later). This is hardly ever true, but it works pretty well in most cases.

Characteristics of NB

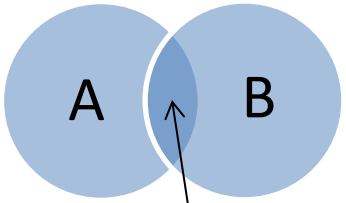
- Learning method: Supervised learning (generative, parametric and a linear model)
- Handling of mixed data types: Yes (both categorical and continuous data)
- Requires linearly separable data? No
- Classification ability: Binary and Multi-class
- Computationally demanding? No
- Ability for online classification: Yes
- Interpretability: High
- When to consider: Text classification, spam detection, target marketing, medical diagnosis, credit approval, recommender systems, emotion detection, face recognition, sentiment analysis

Basic rules of probability

Event: A possible outcome of an observation

P(E) is probability of an event E => E is impossible : P(E)=0

E is certain : P(E)=1



For any event E : $0 \le P(E) \le 1$

All events that are not E: E^c => $P(E) + P(E^c) = 1$

Union of two events A and B: $A \cup B$

Probability of all events in A or B or both: $P(A \cup B)$

Probability of all events common to both A & B: $P(A \cap B)$

Probability of A or B: $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

Zero if A and B have no elements in common (A and B are mutually exclusive)

Intuition behind the Bayes' Theorem

- we can never be fully certain of the world, as it is constantly changing. Fundamental principle behind this theorem, is: update and improve our knowledge of reality (prior) as we get more and more data or evidence.
- Suppose we have an invisible blue ball in the field below

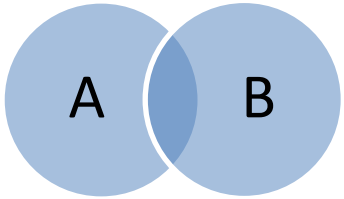
Someone is placing a visible yellow ball and tells you where the blue ball is located with respect to the yellow one, and you update your knowledge for the probability of the location of the blue ball:

0. => Prior : P(x) = 1/16
1. "Left" => Posterior: P(x|1) = 1/8
2. "Behind" => Posterior: P(x|1,2) = 1/4
3. "Front" => Posterior: P(x|1,2,3) = 1/2

Prior gets updated as new evidence is gathered

Conditional Probability

- Probability of A given that B has occurred:



$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \quad \boxed{1} \quad \text{(given that P(B) ≠ 0)}$$

Similarly: $$P(B \mid A) = \frac{P(B \cap A)}{P(A)} \quad \boxed{2} \quad \text{(given that P(A) ≠ 0)}$$

Conditional probability (from 1 & 2): $$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

- Bayes' theorem:

Likelihood of the evidence if the hypothesis is true

Prior probability

$$P(hypothesis \mid data) = \frac{P(data \mid hypothesis)P(hypothesis)}{P(data)}$$

Posterior probability of hypothesis given the evidence

Prior probability that the evidence itself is true

Conditional Probability – cont'd

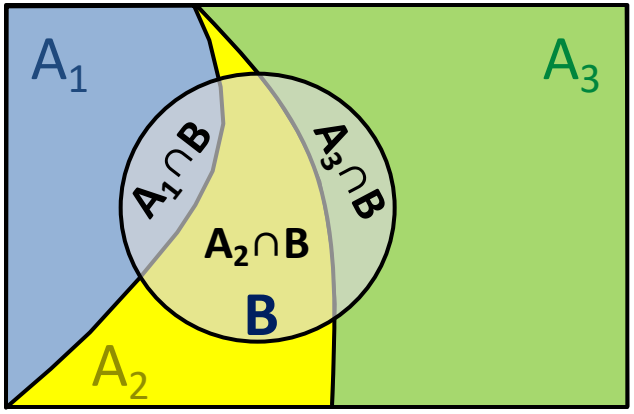
- Expanded form:

$$P(B) = P[(B \cap A) \cup (B \cap A^c)] = P(B \cap A) + P(B \cap A^c)$$

$$= P(B / A)P(A) + P(B / A^c)P(A^c)$$

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \frac{P(B \mid A)P(A)}{\underbrace{P(B \mid A)P(A) + P(B \mid A^c)P(A^c)}_{\sum_j P(B \mid A_j)P(A_j)}}$$

- **Total probability theorem:**

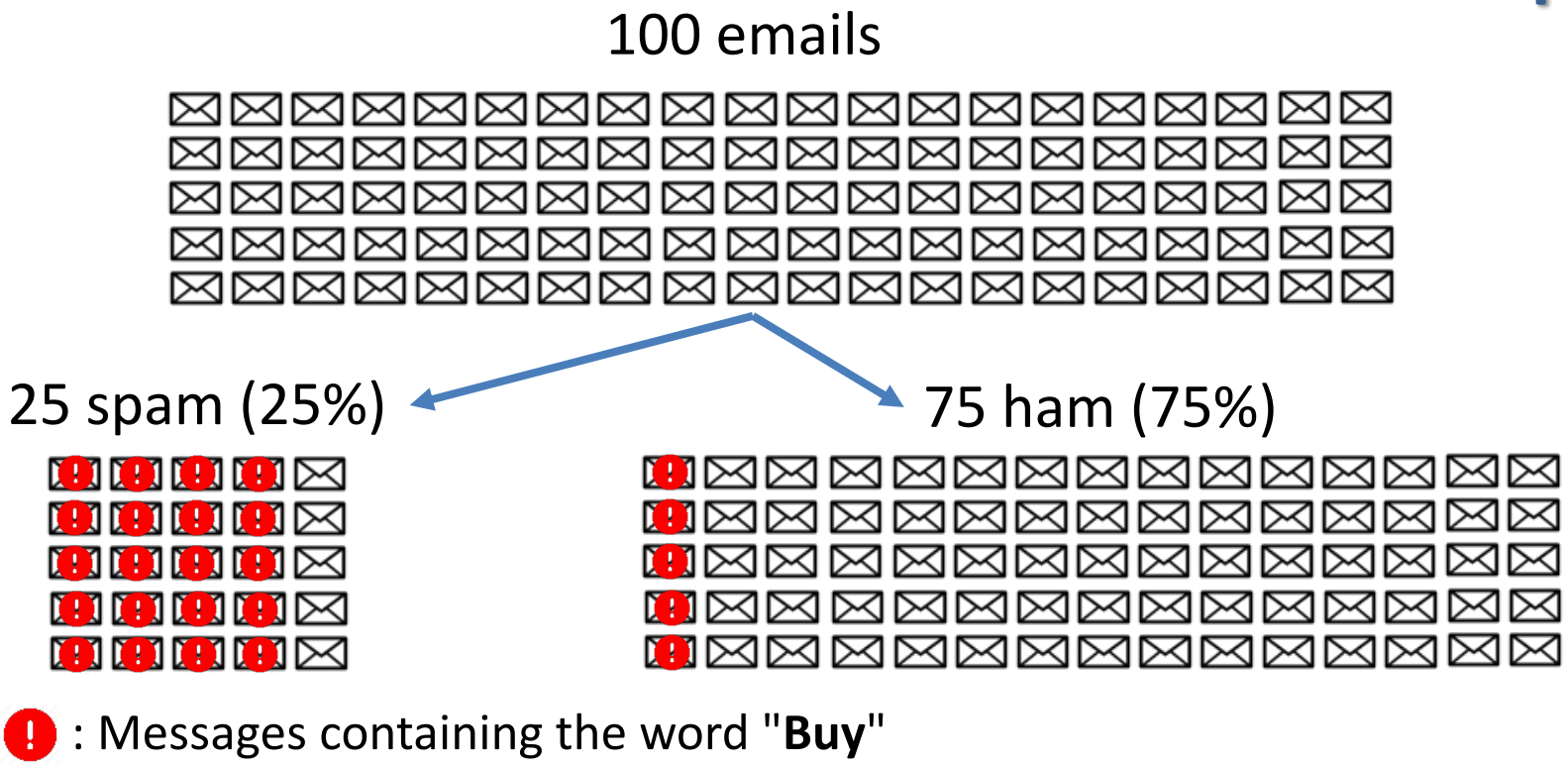


$$P(B) = P(A_1 \cap B) + P(A_2 \cap B) + P(A_3 \cap B)$$

$$P(B) = P(B \mid A_1)P(A_1) + P(B \mid A_2)P(A_2) + P(B \mid A_3)P(A_3)$$

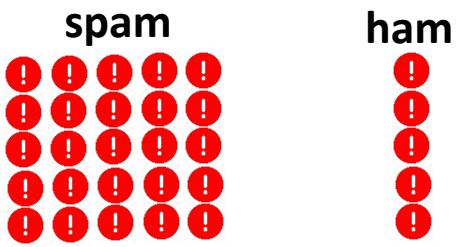
$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{\sum_i P(B \mid A_i)\,P(A_i)}$$

Example I



: Messages containing the word "**Buy**"

- Suppose 20 of the messages labeled "**spam**" contains the word "**Buy**"
- And 5 of the messages labeled "**ham**" contains the word "**Buy**"



If a mail contains the Word "**Buy**", what is the probability that it is **spam**?
20 spam, 5 ham => p = 20/(20+5) = **80%**

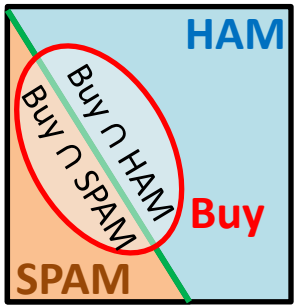
Ref: Naive-Bayes classifier by Luis Serrano

Example I – cont'd

- Let's solve this by using the Bayes' theorem:

$$P(spam \mid Buy) = ?$$

$$P(spam \mid Buy) = \frac{P(Buy \mid spam)P(spam)}{P(Buy)}$$



$$P(Buy) = P(Buy \cap spam) + P(Buy \cap ham)$$

$$P(Buy) = P(Buy \mid spam)P(spam) + P(Buy \mid ham)P(ham)$$

$$P(Buy \mid spam) = \frac{20}{25} \qquad P(spam) = 0.75$$

$$P(Buy \mid ham) = \frac{5}{75} \qquad P(ham) = 0.25$$

$$P(spam \mid Buy) = \frac{\frac{20}{25} * 0.25}{\frac{20}{25} * 0.25 + \frac{5}{75} * 0.75} = \frac{1/5}{1/5 + 5/100} = \frac{0.2}{0.25} = 0.8$$

Example 2

- A new method (VBI: Vision-based Inspection) is deployed to identify defective products. Here is how VBI works:
  - 1% of products are defective (statistical fact)
  - 80% of the VBI identify the defect when present
  - 9.6% of the VBI identify the defect when it's not there
- What's the chance the product is defective when it tests positive?

$$P(D|+) = \frac{P(+|D)P(D)}{P(+)}$$

Total probability theorem

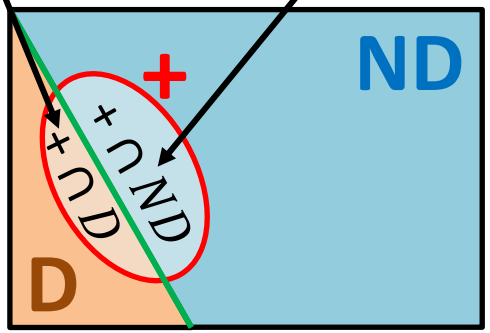
$$P(+) = P(+ \cap D) + P(+ \cap ND)$$
$$= P(+|D)P(D) + P(+|ND)P(ND)$$

$$P(+|D) = \frac{P(+ \cap D)}{P(D)}$$

0.80 0.01

$$P(D|+) = \frac{P(+|D)P(D)}{P(+|D)P(D)+P(+|ND)P(ND)} = 7.8\%$$

0.096 0.99



NB is a generative classifier

- Discriminative classifiers learn the probability of a class given the feature set (remember Logistic regression)

  p(y|x) where y is the class label and x is the feature vector

  – Probability of a certain class given the feature set?

- Generative classifiers learn what kind of features are plausible given the class label

  p(x|y) probability of the feature set given the class label

- Together with the ***p(x|y)***, we also have the ***p(y)***, which is the **class prior** (independent of the features, this is the probability of having the class label in the population)

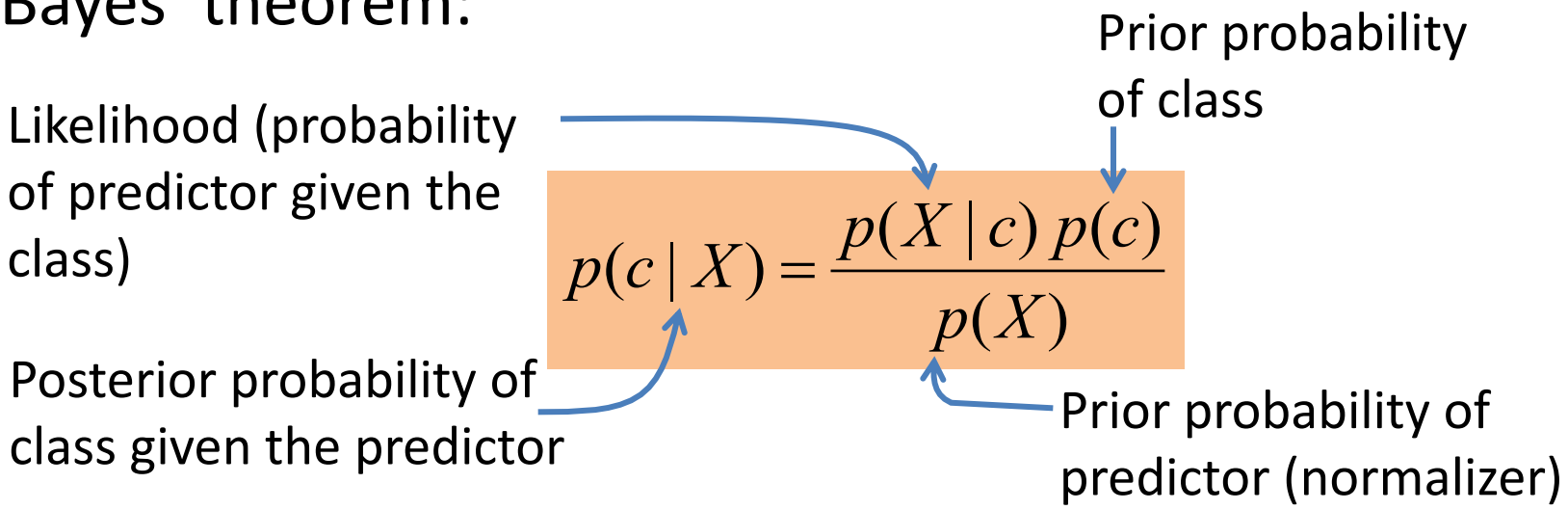
- Example: ***p(y=1)***: probability of carrying the Ebola virus which is supposed to be small. So, the probability of having Ebola virus given the features (set of symptoms):

$$p(y=1 \mid x) = \frac{p(x \mid y=1)\, p(y=1)}{p(x)}$$

$$p(x) = \sum p(x \mid y_i)\, p(y_i)$$
$$= p(x \mid y=1)\, p(y=1) + p(x \mid y=0)\, p(y=0)$$

Why is NB so popular?

- **Classification problem**: What is the probability of class c given the set of features by X? It follows from the Bayes' theorem:



- The numerator is computed as follows:

$$p(X \mid c)p(c) = p(c)p(x_1, x_2, x_3, \ldots x_n \mid c)$$
$$= p(c)p(x_1 \mid c)p(x_2, x_3, \ldots, x_n \mid c, x_1)$$
$$= p(c)p(x_1 \mid c)p(x_2 \mid c, x_1)p(x_3, \ldots x_n \mid c, x_1, x_2)$$
$$= \ldots$$

Why is NB so popular?

- You have to keep track of all probabilities for each possible value of x_i. In the simplest case, x_i is just binary (0 or 1) yielding 2^n probabilities to estimate.
- It is clear that this is way too complex and a simplification is needed. By assuming all features are **conditionally independent**, last equation in the previous slide can be further simplified into the following form:

$$p(X \mid c)p(c) = p(c)\prod_{i=1}^{n} p(x_i \mid c) = p(c)\, p(x_1 \mid c)\, p(x_2 \mid c) \ldots p(x_n \mid c)$$

Probability of class c generating the observed value for feature **1**

Probability of class c generating the observed value for feature **n**

- It all comes down to counting and multiplication which can all be pre-computed. Classification then becomes easy and fast. Assign the class label of whichever has the highest likelihood and it's done!

Types of NB models

- Based on the assumption for distribution of ***p(X|c)***
- **Gaussian NB**
  - Used for continuous data (real numbers), features are assumed to follow a Normal (Gaussian) distribution
- **Bernoulli NB**
  - Used when the feature vectors are binary (male/female) or boolean (True/False) – the binomial model
- **CategoricalNB**
  - Used for categorical features with discrete values (e.g,. ordinal features with 1-2-3 used for cool-warm-hot)
- **Multinomial NB**
  - Used for features containing counts and frequencies (useful for counting how often a word occurs in a text document)

Example

Class	Long	Sweet	Yellow	Total
Banana	**400**	**350**	**450**	500
Orange	**0**	**150**	**300**	300
Other	**100**	**150**	**50**	200
Total	500	650	800	**1000**

We have a training set of 1000 fruits

Based on the features of "long", "sweet" and "yellow", the fruit can belong to one of the following classes: "Banana, "Orange" or "Other"

- New data: {**long, sweet, yellow**} => What is the likelihood of this fruit being any one of the classes?
- **Step 1**: Let's calculate the probability that the fruit is a banana

$$P(Banana \mid Long, Sweet, Yellow) = P(B \mid LSY) = \frac{P(LSY \mid B)P(B)}{P(LSY)} = ?$$

- Assume the features are conditionally independent (given B):

$$P(LSY \mid B) = P(L \mid B)P(S \mid B)P(Y \mid B)$$

(Naive assumption of NB)

- **Step 2**: Let's work out the equations above and plug them in

Example – cont'd

Class	Long	Sweet	Yellow	Total
Banana	**400**	**350**	**450**	500
Orange	**0**	**150**	**300**	300
Other	**100**	**150**	**50**	200
Total	500	650	800	1000

$$P(Long \mid Banana) = 400/500 = 0.8$$
$$P(Sweet \mid Banana) = 350/500 = 0.7$$
$$P(Yellow \mid Banana) = 450/500 = 0.9$$
$$P(Banana) = 500/1000 = 0.5$$

$$P(Banana \mid Long, Sweet, Yellow) = P(B \mid LSY) = \frac{P(LSY \mid B)P(B)}{P(LSY)} = ?$$

$$P(LSY \mid B) = 0.8 * 0.7 * 0.9 = 0.504 \qquad P(B) = 0.5$$

$$P(Banana \mid LSY) = \frac{0.504 * 0.5}{P(LSY)} = \frac{0.252}{P(LSY)} = \frac{0.252}{0.27075} = 93\%$$

$$P(LSY) = P(LSY \mid B)P(B) + P(LSY \mid Orange)P(Orange) + P(LSY \mid Other)P(Other)$$

- Carrying out the same computation for the other two classes:

$$P(Orange \mid LSY) = 0$$
$$P(Other \mid LSY) = \frac{0.019}{P(LSY)} = 7\%$$

Given the current attributes, Naive-Bayes classifies the fruit as Banana

A classification example (2)

- Suppose we're building a tweet classifier which decides whether a tweet is "**positive**" or "**negative**"
- Here is our training data composed of 5 tweets:

tweet	Text	Class
1	I loved the movie	+
2	hated the movie	–
3	great movie, great acting	+
4	a poor script	–
5	bad script, bad movie	–

- Question: Which class does the tweet "**hated the poor acting**" belong to? (**+**) or (**–**)?

A classification example (2) – cont'd

- Since NB is a probabilistic classifier, we need to calculate the probability that the tweet "**hated the poor acting**" is "+" or "–", i.e.,

  P(+ | hated the poor acting) = ?

  P(– | hated the poor acting) = ?

- We need to somehow convert this text into numbers that we can do calculations on. We will
  - treat each tweet as a set of words it contains
  - ignore the word order and sentence construction and use "**word frequencies**"
- Features: Words
- Data: Word frequency (count) for each tweet (row)
- Too simplistic? Yes, but it works surprisingly well!

A classification example (2) – cont'd

- Number of unique words (unigrams) forming the vocabulary **V** : 11

  { I, loved, the, movie, hated, a, great, poor, acting, bad, script }

- We'll now convert the text documents into numerical feature vectors, where features are possible words.
- This process of turning text documents into a feature vector is called "**vectorization**".
- This specific strategy (tokenization, counting and normalization) is called the "**Bag of Words**" representation.

#	I	loved	the	movie	hated	a	great	poor	acting	bad	script	class
1	1	1	1	1		(values are the number of times a word occurs in the given document)						+
2			1	1	1							–
3				1			2		1			+
4						1		1			1	–
5				1						2	1	–

A classification example (2) – cont'd

- In terms of conditional probabilities:

$$P(+ \mid \textit{hated the poor acting}) = \frac{P(\textit{hated the poor acting}|+)P(+)}{P(\textit{hated the poor acting})}$$

$$P(- \mid \textit{hated the poor acting}) = \frac{P(\textit{hated the poor acting}|-)P(-)}{P(\textit{hated the poor acting})}$$

- We just need to compare the numerators for both and take the one with the greater probability for concluding the class of this new text.
- **Problem**: We don't have "hated the poor acting" in our training data.
- **Solution**: Assume every word in a sentence is independent of the other words. Instead of looking at the entire sentence, we'll look at individual words.
- A strong but super useful assumption (thus Naiveness)

A classification example (2) – cont'd

- So we write P(hated the poor acting) as:

  P(hated the poor acting) = P(hated) P(the) P(poor) P(acting)

- Applying this assumption to what we had before:

  P(hated the poor acting|–) = P(hated|–) x P(the|–) x P(poor|–) x P(acting|–)

- All of these individual words actually show up multiple times in our training data and they can be calculated:

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

$$P(\text{hated}|-) = \frac{\text{how many times "}\textbf{hated}\text{" appears in "}-\text{"}}{\text{total number of words in "}-\text{"}} = \frac{P(\textbf{hated} \cap -)}{P(-)} = \frac{1}{10}$$

P(hated the poor acting|–) = P(hated|–) x P(the|–) x P(poor|–) x P(acting|–)

P(hated the poor acting| –) = 0 **?**

doesn't appear in "–"
So, *P(acting|–) = 0*

If one of the terms is zero, the whole expression is nullified!

A classification example (2) – cont'd

- **Solution**? **Laplace smoothing**: Add "1" to every count so that none of the terms is zero. So, add "1" to all possible words in the divisor as well (to ensure a ratio not greater than 1)
- The possible words (entire dictionary of 11) are:

V = { 'I', 'loved', 'the', 'movie', 'hated', 'great', 'a', 'acting', 'poor', 'script', 'bad' }

(1 above 'I'; 11 above 'bad')

- With Laplace Smoothing we get:

word	P(word|+)	P(word|-)
hated	$\frac{0+1}{8+11}$	$\frac{1+1}{10+11}$
the	$\frac{1+1}{8+11}$	$\frac{1+1}{10+11}$
poor	$\frac{0+1}{8+11}$	$\frac{1+1}{10+11}$
acting	$\frac{1+1}{8+11}$	$\frac{0+1}{10+11}$

$$p(w_k \mid +) = \frac{count(w_k \mid +)}{\sum_{w \in V} count(w_k, +)}$$

$$p(w_k \mid +) = \frac{count(w_k \mid +) + 1}{\sum_{w \in V} [count(w_k, +) + 1]}$$

number of times word w_k occurs in class "+"

$$p(w_k \mid +) = \frac{count(w_k \mid +) + 1}{\sum_{w \in V} count(w_k, +) + |V|}$$

number of all words in class "+"

number of unique words in **V**

$(poor \cap +)$

$$p(poor \mid +) = \frac{0+1}{8+11} = 0.0526$$

+1 for "poor"

number of all words in class '+'

+11 for every word in vocabulary

A classification example (2) – cont'd

- We can now multiply all the probabilities:

$P(\text{hated the poor acting}|+) = P(\text{hated}|+) \times P(\text{the}|+) \times P(\text{poor}|+) \times P(\text{acting}|+) = 3.07\ 10^{-5}$

$P(\text{hated the poor acting}|-) = P(\text{hated}|-) \times P(\text{the}|-) \times P(\text{poor}|-) \times P(\text{acting}|-) = 4.11\ 10^{-5}$

tweet

$$P(+|tweet) = \frac{P(tweet|+)\,P(+)}{P(tweet)} = \frac{P(tweet|+)P(+)}{P(tweet|+)P(+) + P(tweet|-)P(-)}$$

2/5

$$= 33.2\%$$

$$P(-|tweet) = \frac{P(tweet|-)\,P(-)}{P(tweet)} = \frac{P(tweet|-)P(-)}{P(tweet|+)P(+) + P(tweet|-)P(-)}$$

3/5

$$= \mathbf{66.8\%}$$ ✓ NB classifies "hated the poor acting" as "—"

NB for continuous data

- NB is designed to work primarily with categorical attributes. In many practical cases, however, variables are often numerical. So how does NB handle numerical attributes?

Temperature	Humidity	Play
85	86	no
89	85	no
80	87	no
90	90	no
86	83	no
83	76	yes
70	88	yes
68	80	yes
64	65	yes
69	70	yes
75	80	yes
75	70	yes
72	90	yes
81	75	yes

Famous "tennis play" data

Method 1: Discretize the continuous variables into different categories using intervals (binning) and turn them into nominal values.

- e.g. temp. smaller than 70 as **cool**, 71-79 as **medium**, above 80 as **hot**
- You can then construct their frequency tables to compute the desired probabilities.
- This can be very subjective and cause loss of information, thus may not be suitable for what you need.

NB for continuous data – cont'd

- **Method 2**: Use probability density functions preserving the continuous values as given. One common practice is to assume Normal distributions for numerical variables

Temp.	Humidity	Play
85	85	no
80	90	no
65	70	no
72	95	no
71	80	no
83	78	yes
70	96	yes
68	80	yes
64	65	yes
69	70	yes
75	80	yes
75	70	yes
72	90	yes
81	75	yes

- Compute the mean and variance from the data set:

  μ_T and σ_T for both "yes"

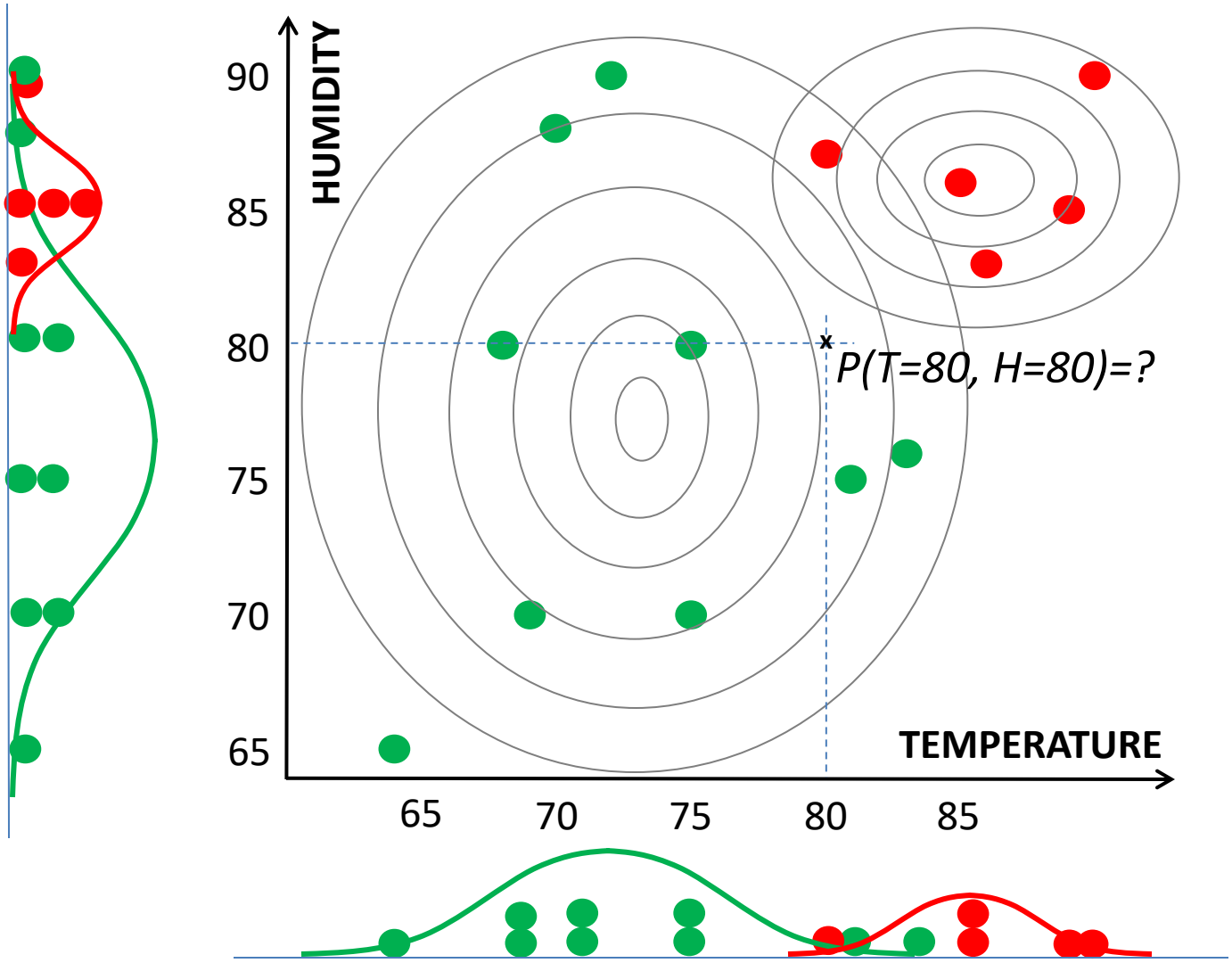
  μ_H and σ_H and "no" classes

- Use the normal distribution to compute probabilities:

$$p(temp = t \mid play = yes) = \frac{1}{\sqrt{2\pi}\sigma_{T,y}} e^{-\frac{(t-\mu_{T,y})^2}{2\sigma_{T,y}^2}}$$

$$p(hum = h \mid play = yes) = \frac{1}{\sqrt{2\pi}\sigma_{H,y}} e^{-\frac{(h-\mu_{H,y})^2}{2\sigma_{H,y}^2}}$$

NB for continuous data – cont'd



NB for continuous data – cont'd

- What is the class (probability) for playing tennis given input x where x = x(temp=T_x, humidity=H_x)?

$$p(H_x \mid yes) = \frac{1}{\sqrt{2\pi}\sigma_{H,y}} e^{-\frac{(H_x-\mu_{H,y})^2}{2\sigma_{H,y}^2}} \qquad p(H_x \mid no) = \frac{1}{\sqrt{2\pi}\sigma_{H,n}} e^{-\frac{(H_x-\mu_{H,n})^2}{2\sigma_{H,n}^2}}$$

$$p(T_x \mid yes) = \frac{1}{\sqrt{2\pi}\sigma_{T,y}} e^{-\frac{(T_x-\mu_{T,y})^2}{2\sigma_{T,y}^2}} \qquad p(T_x \mid no) = \frac{1}{\sqrt{2\pi}\sigma_{T,n}} e^{-\frac{(T_x-\mu_{T,n})^2}{2\sigma_{T,n}^2}}$$

$$p(x \mid yes) = p(H_x \mid yes)\, p(T_x \mid yes) \qquad p(x \mid no) = p(H_x \mid no)\, p(T_x \mid no)$$

$$p(yes \mid x) = \frac{p(x \mid yes)\, p(yes)}{p(x \mid yes)\, p(yes) + p(x \mid no)\, p(no)}$$

$$p(no \mid x) = \frac{p(x \mid no)\, p(no)}{p(x \mid yes)\, p(yes) + p(x \mid no)\, p(no)}$$

NB and missing values

- Handling of missing values with NB is easy: Just skip it!
- We can simply ignore the instance for which the value is missing. Remember the expression for NB:

$$p(X_1, X_2, \ldots, X_j, \ldots X_n \mid y) = \prod_{i \neq j}^{n} P(X_i \mid y)$$

Product goes over all attributes except "j", the one that's skipped (missing)

Tossing a coin three times: E={ X_1=Head, X_2=missing, X_3=Tail }
p(E) = p(H,**H**,T) + p(H,**T**,T) as the missing value is either **H** or **T**
P(E) = 1/8 + 1/8 = 1/4 (same as p(H,T) = 1/4)

$$p(X_1, \ldots, X_j, \ldots, X_n \mid y) = P(X_1 \mid y) \ldots P(X_j \mid y) \ldots P(X_n \mid y)$$

$$\sum_{X_j} p(X_1, \ldots, X_j, \ldots X_n \mid y) = \sum_{X_j} P(X_1 \mid y) \ldots P(X_j \mid y) \ldots P(X_n \mid y)$$

over all possible values of X_j

$$= P(X_1 \mid y) \ldots \underbrace{\sum_{X_j} P(X_j \mid y)}_{1} \ldots P(X_n \mid y) = \prod_{i \neq j}^{n} p(X_i \mid y)$$

Problems with NB – cont'd

- **Zero Frequency (zero observation) problem**
  - When a categorical feature has a value in the test set, but not in the training set, the model assigns a 0 probability and is unable to make a prediction.
  - To fix it, we apply a smoothing technique (Laplace correction) and add 1 to every feature. This is essentially assigning an arbitrarily low probability to that class to make it different from zero.
- **Not much "fine-tuning"**
  - NB classifiers have very limited options for parameter tuning. So, it's best to focus on data pre-processing and feature selection.
  - In case overfitting occurs, one can tweak the priors.

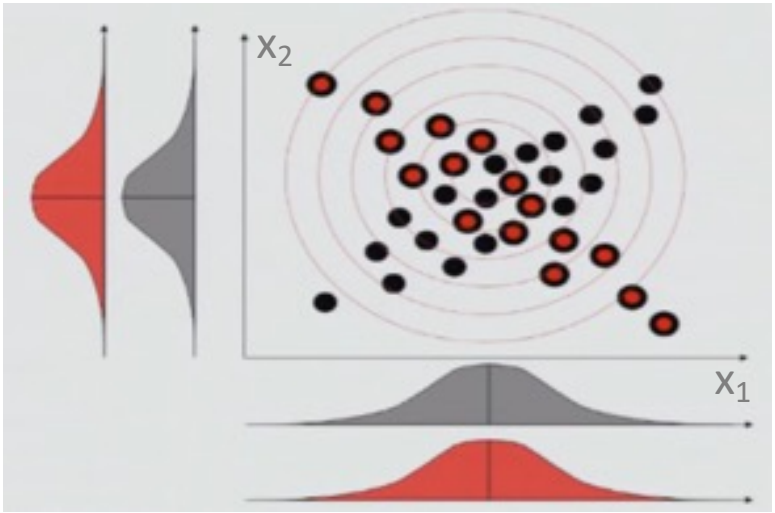
Problems with NB – cont'd

- **No combination of classifiers**
  - Combination of classifiers (like ensembling, bagging and boosting) won't help with NB as the purpose with these techniques is to reduce the variance. NB has no variance to minimize (no loss function to minimize).
  - NB employs a simple hypothesis function with no dependencies in features (unable to model feature interactions unlike many other classifiers), hence, it's a high-bias/low-variance model that is unable to handle complex problems.
- **Deviations from normality**
  - For continuous features that don't follow a Normal distribution, try variable transformation.

Problems with NB – cont'd

- **Inversely correlated attributes**
  - Once you project all data points on both axes, distributions for 2 classes are on top of each other (almost identical) providing no extra prediction power. So, NB can't distinguish between the classes. What differentiates these classes isn't the means or variances. It's the covariance . For black class, x_1 and x_2 are positively correlated, for red class, they are negatively correlated. This is the only difference between the classes. NB cannot deal with correlation. It assumes conditional independence (naive). If the only thing that separates the 2 classes is the way in which attributes are correlated, you're out of luck.

Victor Lavrenko IAML5.9 - Gaussian Naive Bayes classifier



Problems with NB – cont'd

- **Double-counting effect (highly-correlated features)**
  - When both features encode essentially the same information, the classifier "double-counts" the effect of highly correlated features (voting them twice overinflates their importance), and gets pushed closer to a given label than is justified. This is particularly true for large and sparse data (text mining).
  - Of course, we don't usually build NB classifiers that contain two identical features. However, we do build classifiers that contain features which are dependent on one another. For example, the features ends-with **(a)** and ends-with **(vowel)** are dependent on each other. Because if an input value has the first feature, then it must also have the second feature. For features like these, the duplicated information may be given more weight than is justified by the training set.

Problems with NB – cont'd

- **Presence of mixed data (continuous + categorical)**
  - GaussianNB (continuous), BernoulliNB (binary), and MultinomialNB (nominal) are employed for different types of data. How to proceed when we have mixed data?
  - A few approaches (see the Notebook for some examples):
  - Use binning with intervals that contain roughly equal number of data points for continuous variables and turn them into categorical variables (loss of information)
  - Compute the probabilities for each feature type independently (due to nature of the underlying assumption of NB) and multiply them together (by taking care of normalization)
  - Fit different models for different features and then transform the data set by taking the class assignment probabilities as new features, and then refitting a new model (say GaussianNB) on the new features.

Improving NB

- **Use log probabilities**
  - Probabilities can be small numbers and often lead to inaccurate results (due to precision of floating-point values) when multiplied together. It is best to work in the logarithmic realm and add them up rather than multiply them. You can convert it back to original later.
- **Remove redundant features**
  - Due to possible effect of double-counting of correlated features, you can remove one of the features.
- **One vs Rest approach**
  - For features that have many categorical levels, you can try a two-class feature (category A vs all-else) and keep experimenting with different combinations

Pros and Cons

- **Pros**
  - A **very good choice for a small amount of training data** (as the estimates are based on the joint density function)
  - Reliably a high bias/low variance classifier. Try for problems with noisy data, lots of features, and few samples
  - Despite its simplicity, surprisingly accurate
  - Easy, **very fast to train** (single scan) and classify
  - **Not sensitive to irrelevant features**
  - Works with both categorical and continuous data (requires work)
  - Handles multiple classes
  - **Tolerant to missing values** (not implemented in scikit-learn)
  - Can also be **used to generate instances of the problem**
- **Cons**
  - Assumes independence of features (cannot learn interactions between features – fits feature weights independently)
  - Suffers from multi-collinearity
  - Sensitive to how input is prepared

Naive-Bayes w/ Python

```
# using scikit-learn library
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

Clf = GaussianNB()
Clf.fit(X_train, y_train)
accuracy_score(y_test, Clf.predict(X_test))
```

Addendum

Bias-Variance property of NB

- **Why is NB a high-bias/low variance classifier?**
- The Naive Bayes classifier employs a very simple (linear) hypothesis function to model data. It suffers from high bias, or error resulting from inaccuracies in its hypothesis class, because its hypothesis function is so simple it cannot accurately represent many complex situations.
- On the other hand, it exhibits low variance, or failure to generalize to unseen data based on its training set, because its hypothesis class' simplicity prevents it from overfitting to its training data.

Is NB a linear classifier?

- **Is NB a linear classifier? What is a decision boundary like?**
  - Suppose we have two populations whose distributions are Gaussian: $N(\mu_0, \sigma_0^2)$ and $N(\mu_1, \sigma_1^2)$
  - Want to classify whether a person is female (c=0) or male (c=1) based on height

$$p(x \mid c=0) = \frac{1}{\sqrt{2\pi}\sigma_0} e^{-\frac{(x-\mu_0)^2}{2\sigma_0^2}} \qquad p(x \mid c=1) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}}$$

  - What we need is:

$$\hat{c}(x) = ?$$

$$\hat{c}(x) = 1 \;\; if \;\; p(c=1 \mid x) > 0.5; \quad \hat{c}(x) = 0 \;\; otherwise$$

  - We'll assume equal probability of being male or female

$$p(c=1) = 0.5$$

  - This is the condition that will form the decision boundary

Is NB a linear classifier?

– Probability of class=male given the feature:

$$p(c=1 \mid x) \propto p(x \mid c=1)\, p(c=1)$$

– Let's assume $p(c=1) = p_1$

$$\log\left[p(x \mid c=1)\, p(c=1)\right] = \log\left[\frac{1}{\sqrt{2\pi}\sigma_1}\right] - \frac{1}{2{\sigma_1}^2}(x-\mu_1)^2 + \log p_1$$

$$\log\left[p(x \mid c=0)\, p(c=0)\right] = \log\left[\frac{1}{\sqrt{2\pi}\sigma_0}\right] - \frac{1}{2{\sigma_0}^2}(x-\mu_0)^2 + \log p_0$$

– The decision boundary between any two classes is where the log of the ratio (log odds) is zero (why?)

$$\log \frac{p(x \mid c=1)\, p(c=1)}{p(x \mid c=0)\, p(c=0)} = -\frac{1}{2{\sigma_1}^2}(x^2 - 2\mu_1 x) + \frac{1}{2{\sigma_0}^2}(x^2 - 2\mu_0 x)$$

constant

$$-\frac{{\mu_1}^2}{2{\sigma_1}^2} + \frac{{\mu_0}^2}{2{\sigma_0}^2} + \log\left[\frac{1}{\sqrt{2\pi}\sigma_1}\right] - \log\left[\frac{1}{\sqrt{2\pi}\sigma_0}\right] + \log p_1 - \log p_0$$

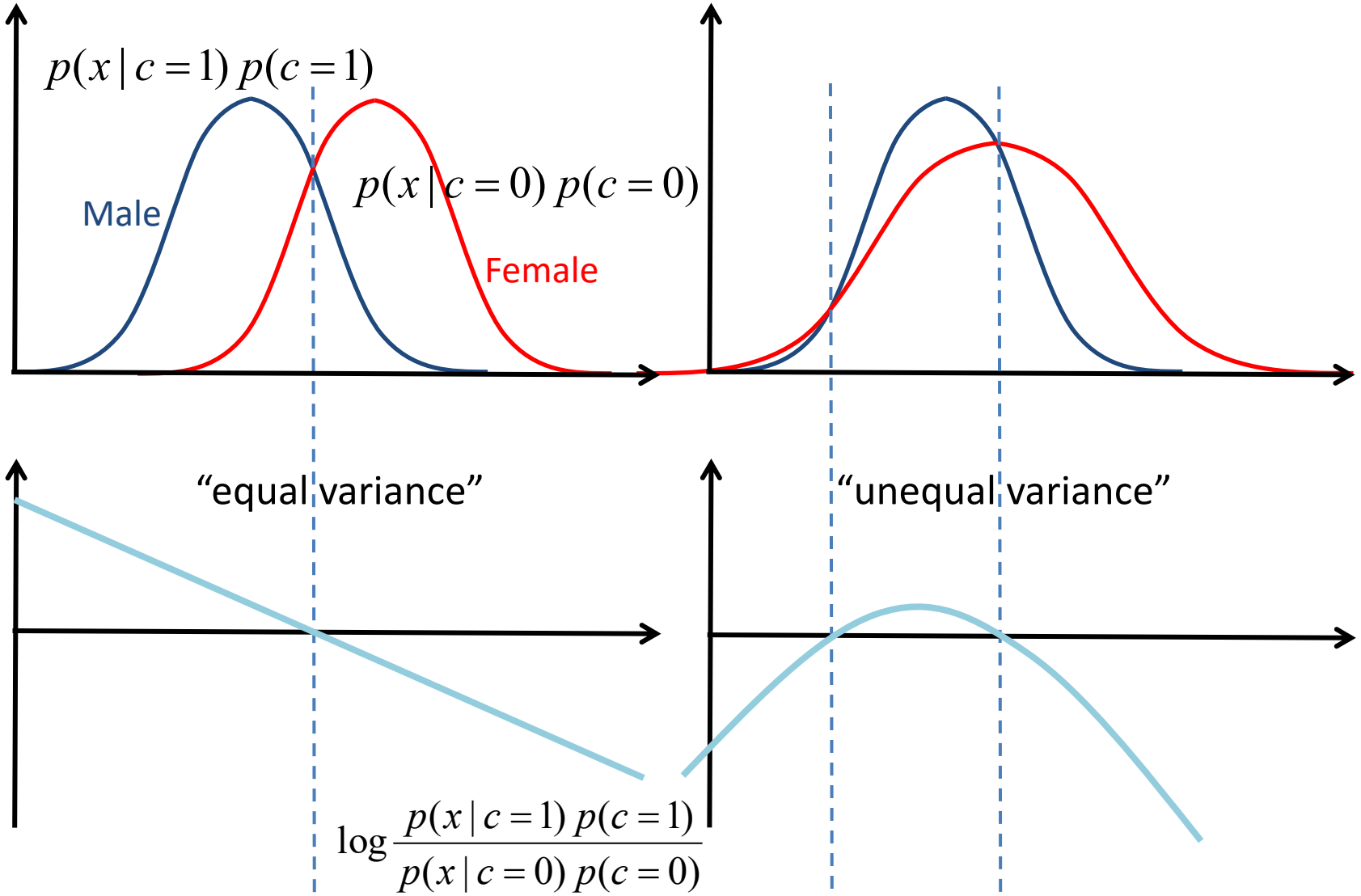
Is NB a linear classifier?

- So the log of the ratio becomes:

$$\log \frac{p(x \mid c=1)\, p(c=1)}{p(x \mid c=0)\, p(c=0)} = \frac{1}{2}\left(\frac{1}{\sigma_0^{\,2}} - \frac{1}{\sigma_1^{\,2}}\right)x^2 + \left(\frac{\mu_1}{\sigma_1^{\,2}} - \frac{\mu_0}{\sigma_0^{\,2}}\right)x + \textbf{constant}$$

  - If the data in each class has a Gaussian density with **unequal variance**, then the boundary between any two classes is a **quadratic function** of x.
  - If the data in each class has a Gaussian density with **equal variance**, the coefficient of the quadratic term disappears and the boundary between any two classes becomes a **line**.
  - Summary for the decision boundary:
    - General case: parabolic curve
    - Different means, same variance: Line (or plane)
    - Same mean, different variance: circle / ellipse
  - See the next slide for graphical representations

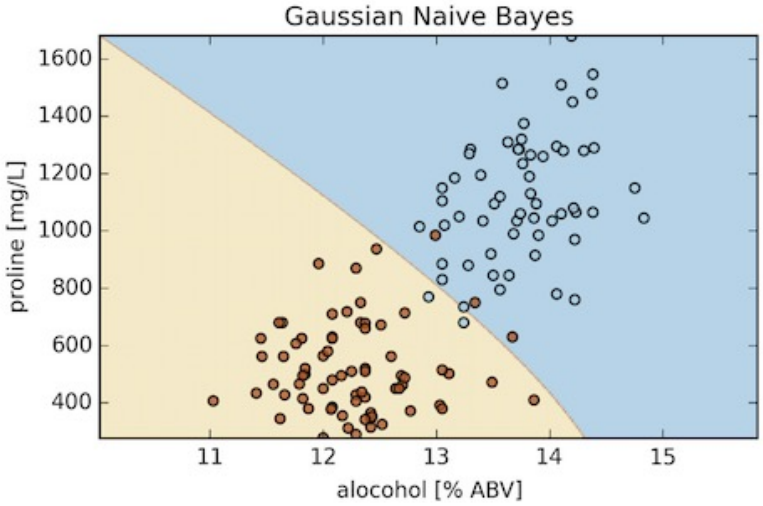
Is NB a linear classifier?



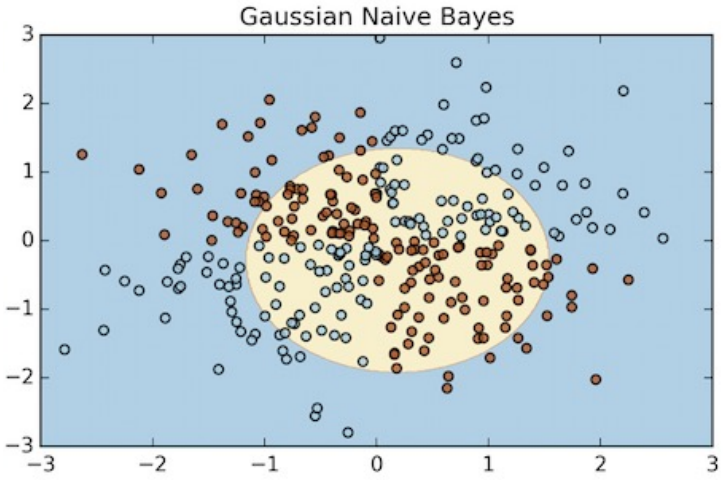
Is NB a linear classifier?

- So, it's a (piecewise) quadratic decision boundary for the Gaussian model. The multinomial model has a linear boundary. Here are some plots taken from S. Raschka:

UCI Wine data set



An XOR toy data set



Ref: https://www.quora.com/What-is-the-decision-boundary-for-Naive-Bayes

Mixed data in NB

- How do to deal with mixed (categorical + continuous) data in NB?
- There are 3 types of formulations for NB in scikit-learn:
    1. BernoulliNB (binary features)
    2. MultinomialNB (nominal features)
    3. GaussianNB (continuous features)
- In the presence of mixed data, you cannot mix the features. We have a few options:
- **1.** Convert your continuous data into a categorical representation by computing percentiles for each continuous variable and then bin those percentiles as categorical levels. For example, you can take the temperature for and create the following bins: "cold", "cool", "normal", "warm", "hot" ensuring that each bin contains approximately 20% of the population of your training set (not recommended as discretization will throw away useful information contained in a continuously varying data)

Mixed data in NB – cont'd

- **2.** You can independently fit a GaussianNB model on the continuous part of the data and a MultinomialNB (BernoulliNB depending on the nature of the categorical data) model on the categorical part. Then transform all the dataset by taking the class assignment probabilities (with predict_proba method) as new features as can be seen from the command down below. You can then easily refit a new model (e.g. a new GaussianNB) on these new features.

```
np.hstack((multinomial_probas, gaussian_probas))
```

Ref: https://stackoverflow.com/questions/14254203/mixing-categorial-and-continuous-data-in-naive-bayes- \
classifier-using-scikit-lea

- **3.** The last option is to find the individual probability values (as in 2) for both the continuous and categorical features. Let's assume that we have two features (feature1 is continuous, and feature2 is categorical). Here is the probability we need to calculate:

Mixed data in NB – cont'd

- Example data set

feature1	feature2	class
126	low	c1
220	low	c2
191	medium	c1
128	high	c1
125	low	c2
...	...	..

$$p(c_1 \mid f_1, f_2) = \frac{p(f_1, f_2 \mid c_1)p(c_1)}{p(f_1, f_2)}$$

$$p(c_1 \mid f_1, f_2) = \frac{p(f_1 \mid c_1)p(f_2 \mid c_1)p(c_1)}{p(f_1, f_2 \mid c_1)p(c_1) + p(f_1, f_2 \mid c_2)p(c_2)} = \frac{p(f_1 \mid c_1)p(f_2 \mid c_1)p(c_1)}{D}$$

Similarly:

$$p(c_2 \mid f_1, f_2) = \frac{p(f_1 \mid c_2)p(f_2 \mid c_2)p(c_2)}{D}$$

Mixed data in NB – cont'd

- By using the GaussianNB on $feature1$, and MultinomialNB on $feature2$ based on the target values, we get the probabilities (predict_proba method) separately:

$$p(c_1 \mid f_1) = \frac{p(f_1 \mid c_1)p(c_1)}{p(f_1 \mid c_1)p(c_1) + p(f_1 \mid c_2)p(c_2)} = \frac{p(f_1 \mid c_1)p(c_1)}{P_1} = p_{11} \Rightarrow p(f_1 \mid c_1) = \frac{p_{11}}{p(c_1)} P_1$$

$$p(c_1 \mid f_2) = \frac{p(f_2 \mid c_1)p(c_1)}{p(f_2 \mid c_1)p(c_1) + p(f_2 \mid c_2)p(c_2)} = \frac{p(f_2 \mid c_1)p(c_1)}{P_2} = p_{12} \Rightarrow p(f_2 \mid c_1) = \frac{p_{12}}{p(c_1)} P_2$$

$$p(c_2 \mid f_1) = \frac{p(f_1 \mid c_2)p(c_2)}{p(f_1 \mid c_1)p(c_1) + p(f_1 \mid c_2)p(c_2)} = \frac{p(f_1 \mid c_2)p(c_2)}{P_1} = p_{21} \Rightarrow p(f_1 \mid c_2) = \frac{p_{21}}{p(c_2)} P_1$$

$$p(c_2 \mid f_2) = \frac{p(f_2 \mid c_2)p(c_2)}{p(f_2 \mid c_1)p(c_1) + p(f_2 \mid c_2)p(c_2)} = \frac{p(f_2 \mid c_2)p(c_2)}{P_2} = p_{22} \Rightarrow p(f_2 \mid c_2) = \frac{p_{22}}{p(c_2)} P_2$$

- Now going back to $p(c1|f1,f2)$ and $p(c2|f1,f2)$ we find:

$$p(c_1 \mid f_1, f_2) = \frac{p(f_1 \mid c_1)p(f_2 \mid c_1)p(c_1)}{D} = \frac{P_1 P_2}{D} \frac{p_{11} p_{12}}{p(c_1)}$$, and

$$p(c_2 \mid f_1, f_2) = \frac{p(f_1 \mid c_2)p(f_2 \mid c_2)p(c_2)}{D} = \frac{P_1 P_2}{D} \frac{p_{21} p_{22}}{p(c_2)}$$

Mixed data in NB – cont'd

- Normalizing probabilities we get:

$$p(c_1 \mid f_1, f_2) = \frac{p(c_1 \mid f_1, f_2)}{p(c_1 \mid f_1, f_2) + p(c_2 \mid f_1, f_2)} = \frac{\frac{p_{11} p_{12}}{p(c_1)}}{\frac{p_{11} p_{12}}{p(c_1)} + \frac{p_{21} p_{22}}{p(c_2)}}$$

$$p(c_2 \mid f_1, f_2) = \frac{p(c_2 \mid f_1, f_2)}{p(c_1 \mid f_1, f_2) + p(c_2 \mid f_1, f_2)} = \frac{\frac{p_{21} p_{22}}{p(c_2)}}{\frac{p_{11} p_{12}}{p(c_1)} + \frac{p_{21} p_{22}}{p(c_2)}}$$

- where p_1 and p_2 are the priors for the target variable and can easily be calculated from the training data. p_{11}, p_{12}, p_{21}, and p_{22} are the probabilities computed by fitting a model to each feature (continuous or categorical) using the relevant NB model from scikit-learn library and then executing the "predict_proba" command separately.