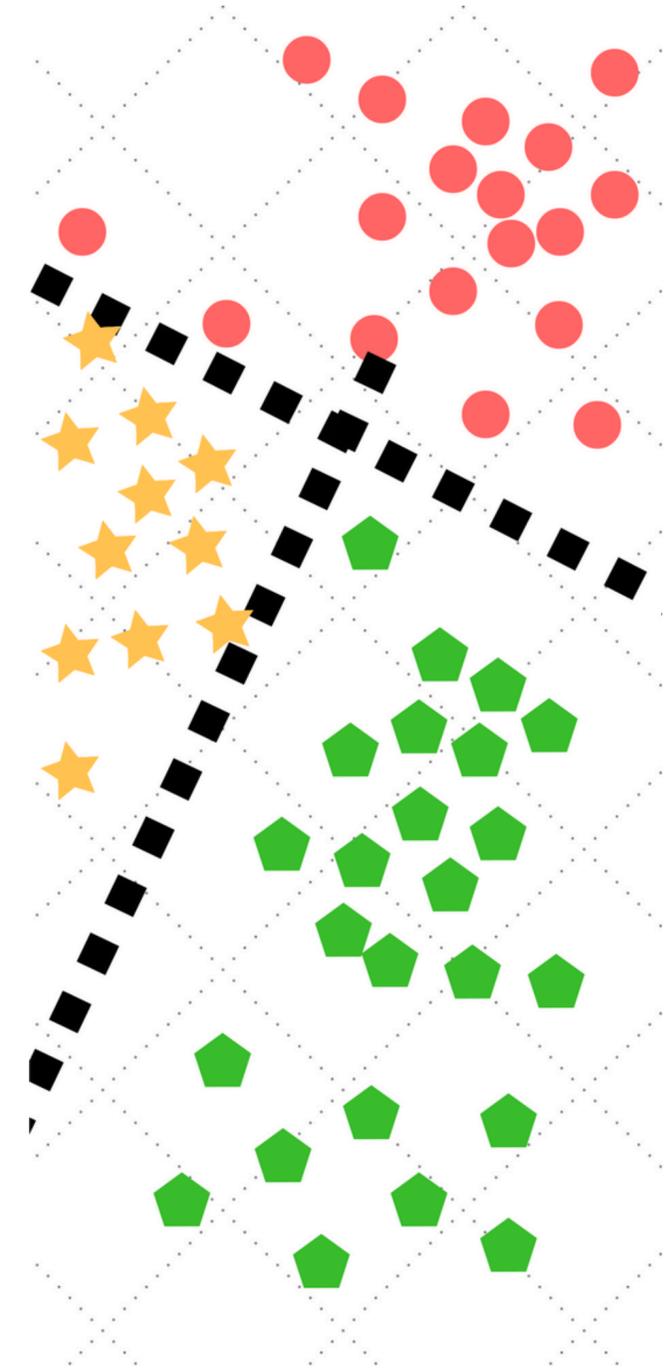# ECE 20875
Python for Data Science

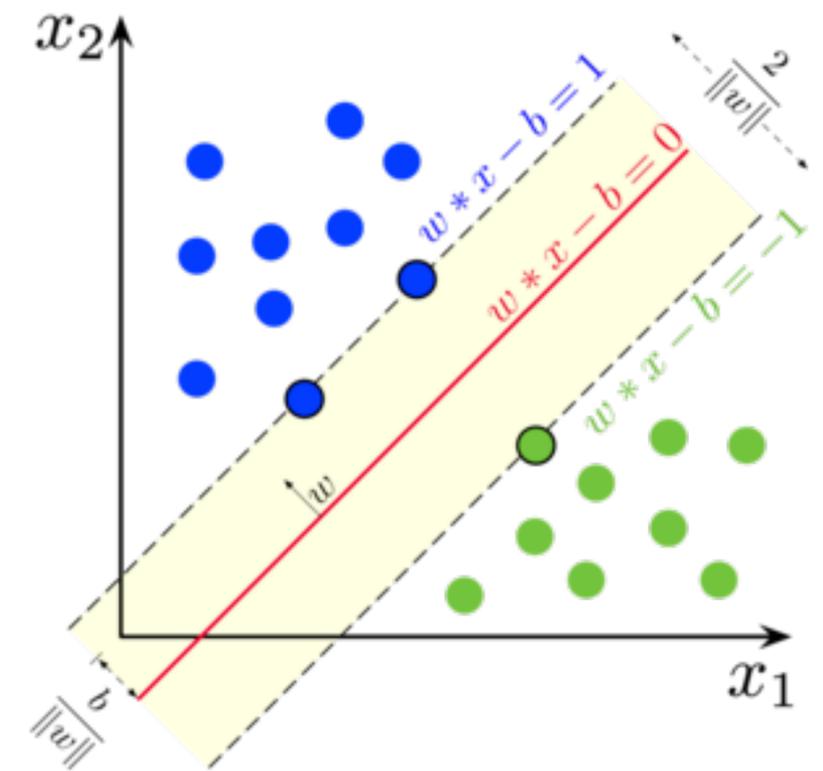**Chris Brinton and David Inouye**

classification: naive bayes

what is classification?

- Given a data point, tell me what class it falls into

  - Is this animal a mammal, a bird, a fish, …

  - Is this picture a cat, a horse, a car, …

- Generally, we want to *learn* a classifier

  - We are given a bunch of data points: x_1, x_2, x_3, etc. where each data point is **labeled** with its class

  - Classifier should be able to tell which class a new datapoint belongs to

  - Because we start with labeled data, this is another example of supervised learning

defining a classifier formally

- Consider datapoints in a d-dimensional space, i.e., where each datapoint i is defined by d features, written as $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$.

- Each datapoint i is from one of K possible **classes** $C_0, C_1, \ldots, C_{K-1}$. We denote i's class as $y_i \in \{C_0, \ldots, C_{K-1}\}$.

- A **classifier** is a function that predicts a datapoint's class. Formally, $\hat{y}_i = f(\mathbf{x}_i)$, where $\hat{y}_i \in \{C_0, \ldots, C_{K-1}\}$ is the prediction of the actual class y_i.

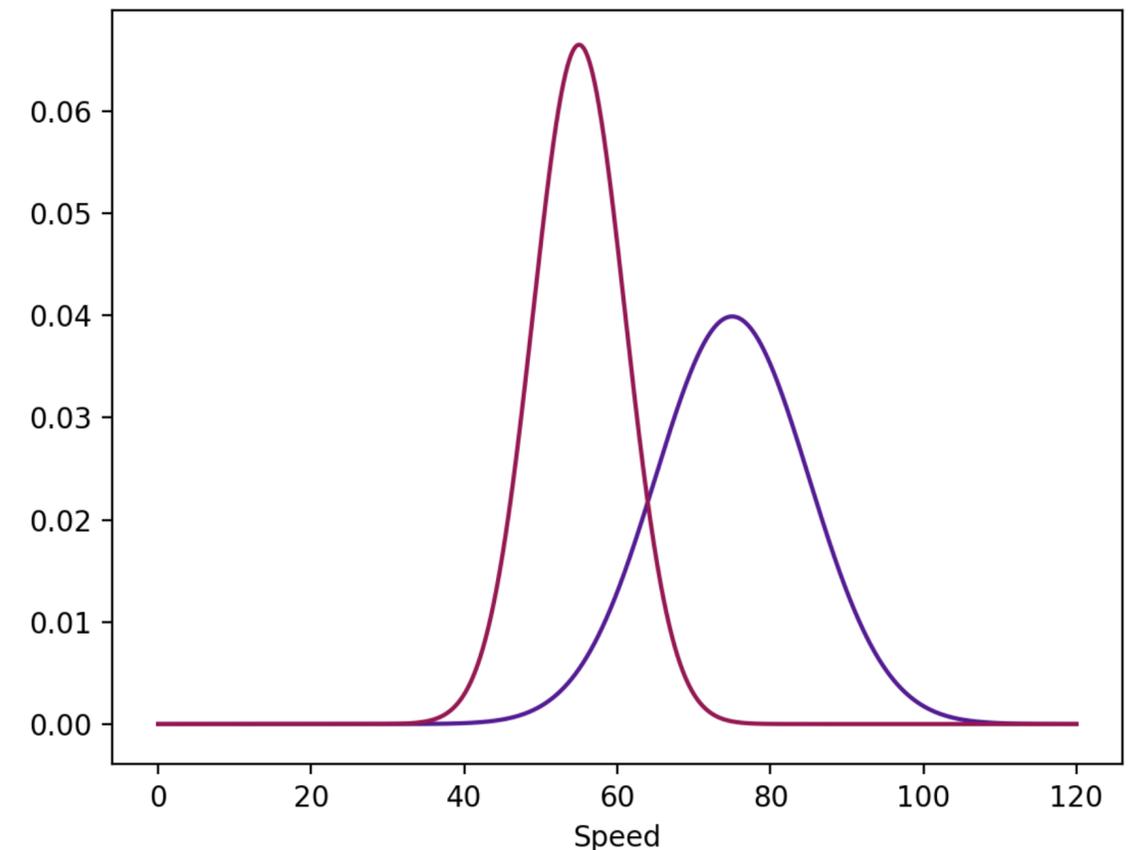- We will focus on learning classifiers for two classes, i.e., $K = 2$

four different methodologies

- Like regression, there are many different flavors of classification algorithms

- We will talk about four different, yet common and representative methodologies:

  1. **Naïve Bayes**: Simple model to use. But it's *parametric* — requires assumptions about data.

  2. **k-nearest neighbor (kNN)**: Very easy model to understand. Expensive model to evaluate. But it's *non-parametric* — requires few assumptions about data.

  3. **Logistic regression**: Another non-parametric model (derived from linear regression) which is harder to interpret than **kNN** but can have more explanatory power.

  4. **Neural networks**: Trendy approach! Essentially cascading nonlinear functions together to maximize potential predictive power.
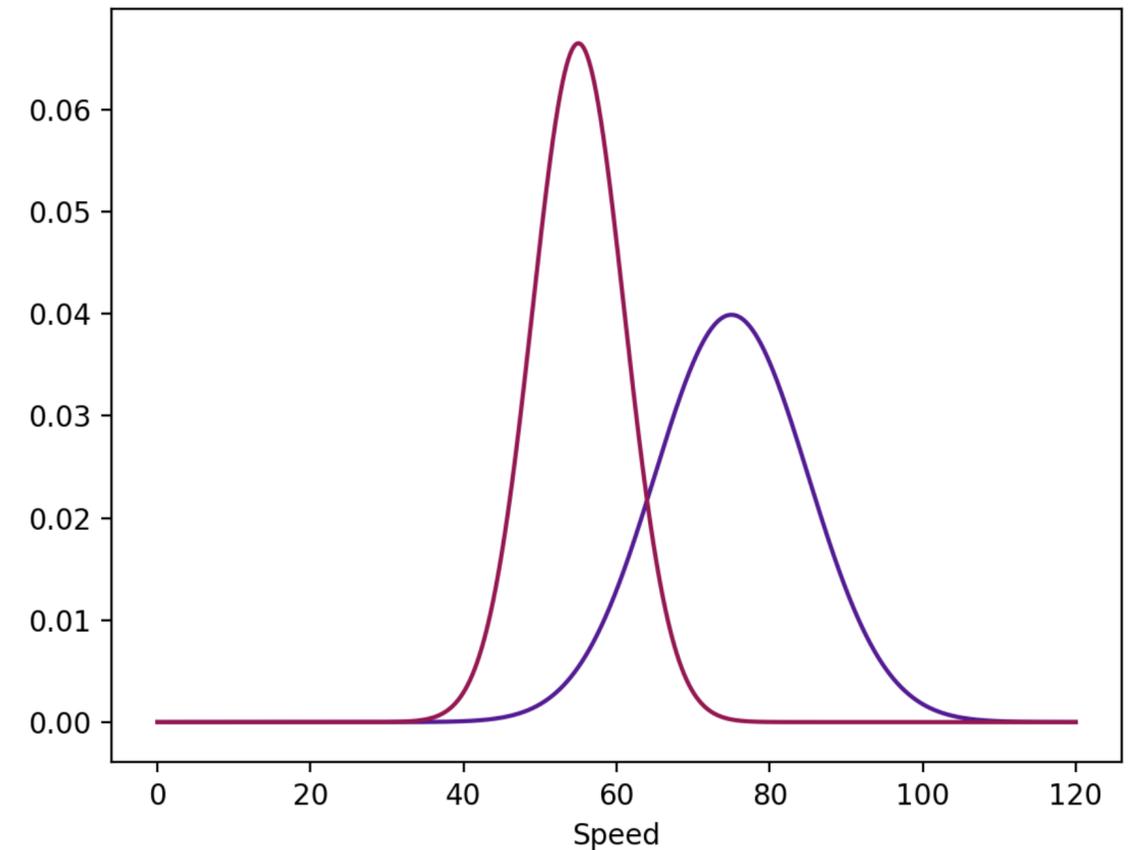
naïve bayes model

- Basic idea: Each class of data can be described by a probability distribution showing how likely different data points are

  - If I have a new data point x_i, which distribution is it more likely to come from?

- Example: Two classes of cars — **sports cars** C_1 and **minivans** C_0

  - Each car can be described by its average speed x, and the two classes C_0, C_1 have different distributions of speeds (i.e., sports cars have a higher average speed than minivans)

  - I see a new speed reading x_i. Is car i a **sports car** (i.e., $y_i = C_1$) or a **minivan** (i.e., $y_i = C_0$)?
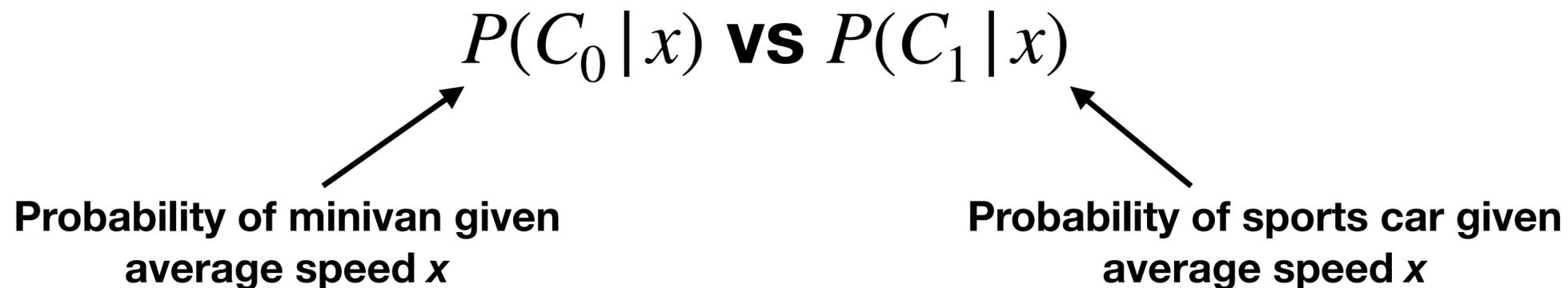
naïve bayes model

- Similar to Gaussian mixture models, we can define our naïve bayes model according to the following probabilities/distributions:

    - $P(C_0), P(C_1)$ - **Prior probabilities** of classes 0 and 1 (in our example, probability of **sports car** and **minivan**)

    - $P(x|C_0)$ - Distribution for class 0 (e.g., distribution of **minivan**)

    - $P(x|C_1)$ - Distribution for class 1 (e.g., distribution of **sports car**)

predicting with naïve bayes

- Basic idea: Each class of data can be described by a probability distribution showing how likely different data points are

  - If I have a new data point x, which distribution is it **more likely** to have been drawn from?

- Formally, this is quantified in terms of the **posterior distributions** (conditional probabilities):

$$P(C_0 \mid x) \textbf{ vs } P(C_1 \mid x)$$

Probability of minivan given average speed *x*

Probability of sports car given average speed *x*

- Problem: How do we actually infer (or compute) these probabilities given our model?

bayes' theorem to the rescue

- Bayes' theorem allows us to compute the posterior probability (which we do not know) using quantities from the naïve bayes model (which we can estimate)

- We have seen this "trick" before, most recently with Gaussian mixture models. For the case of comparing

$$P(C_0 \,|\, x) \textbf{ vs } P(C_1 \,|\, x)$$

we can write

$$P(C_0 \,|\, x) = \frac{P(x \,|\, C_0)P(C_0)}{P(x)} \qquad\qquad P(C_1 \,|\, x) = \frac{P(x \,|\, C_1)P(C_1)}{P(x)}$$

bayes' theorem to the rescue
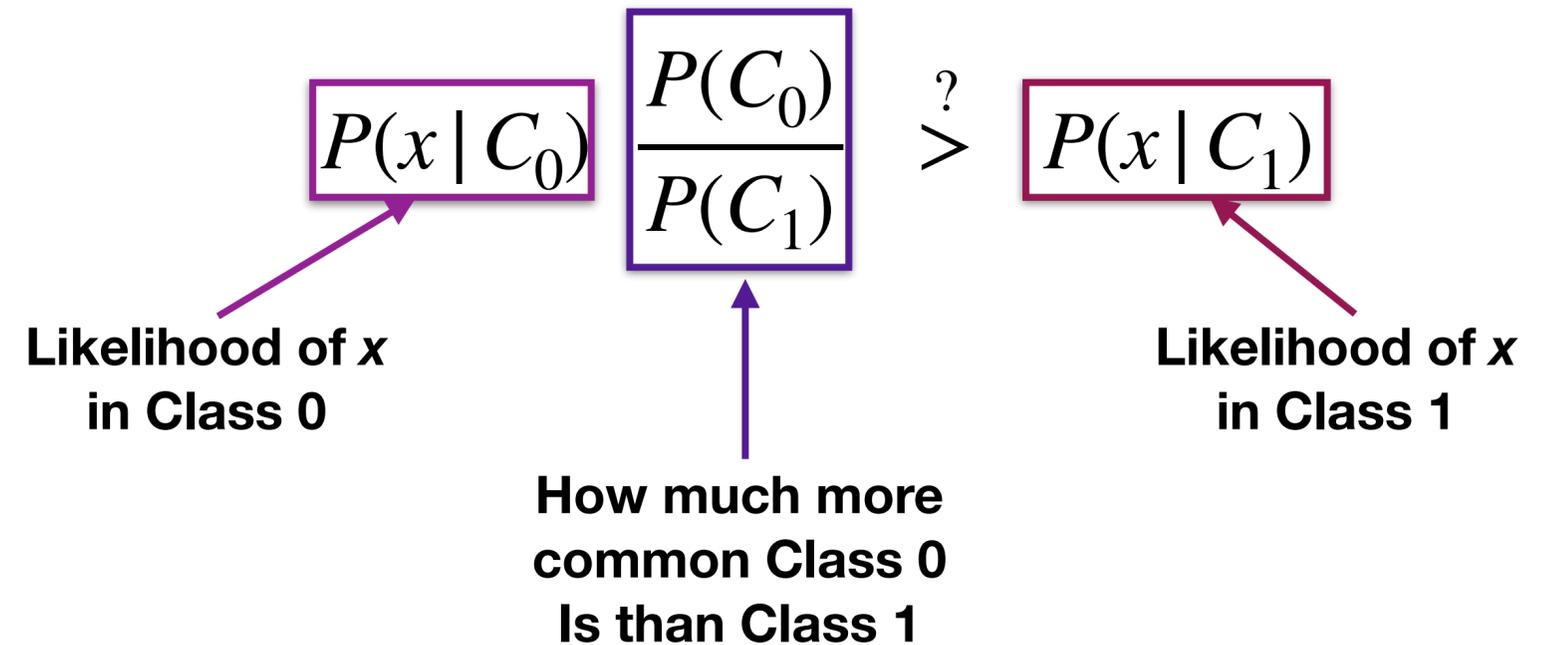
- With some simple algebra, we can write the following:

$$P(C_0 \mid x) \overset{?}{>} P(C_1 \mid x)$$

$$\frac{P(x \mid C_0)P(C_0)}{P(x)} \overset{?}{>} \frac{P(x \mid C_1)P(C_1)}{P(x)}$$

$$P(x \mid C_0)P(C_0) \overset{?}{>} P(x \mid C_1)P(C_1)$$

$$P(x \mid C_0)\frac{P(C_0)}{P(C_1)} \overset{?}{>} P(x \mid C_1)$$

- So our comparison boils down three quantities:

$$\boxed{P(x \mid C_0)} \boxed{\frac{P(C_0)}{P(C_1)}} \overset{?}{>} \boxed{P(x \mid C_1)}$$

Likelihood of *x* in Class 0

How much more common Class 0 Is than Class 1

Likelihood of *x* in Class 1
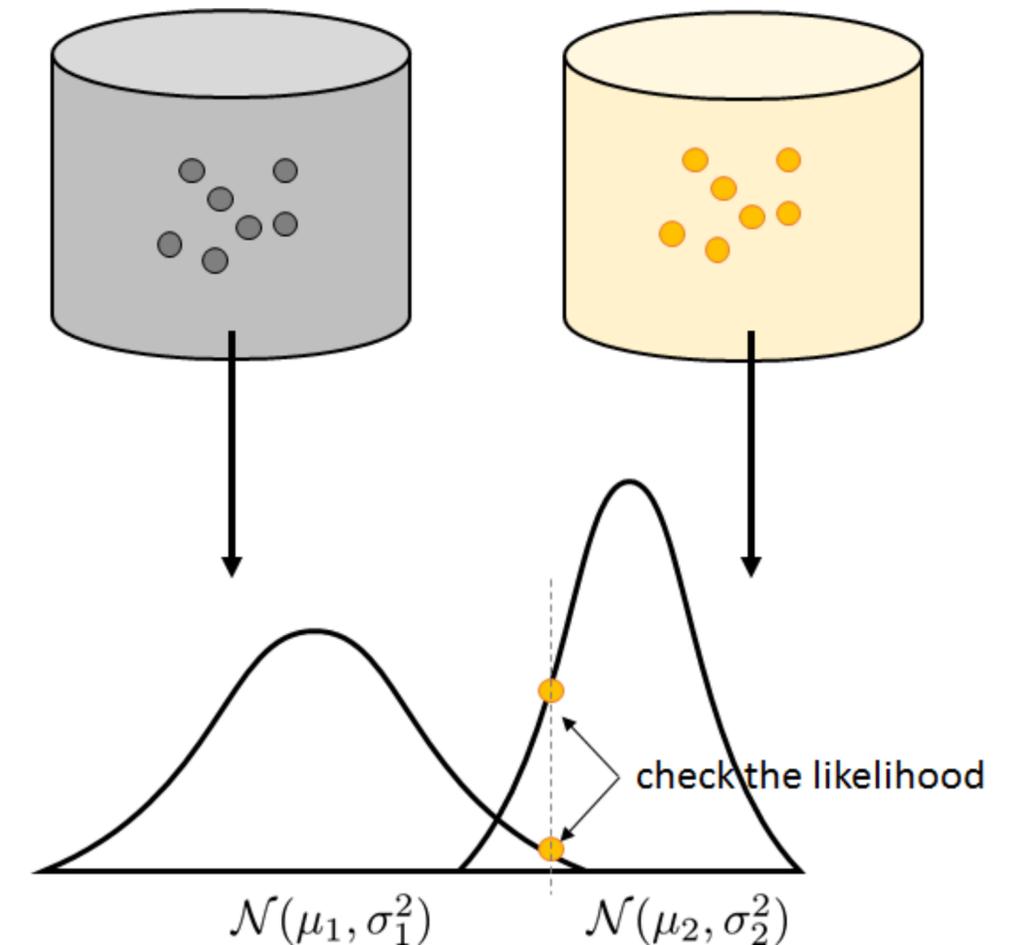
estimating the naïve bayes model

- Once we've applied Bayes' theorem, we can estimate the necessary probabilities

- $P(C_0)/P(C_1)$: Comes from either *prior knowledge* or can be estimated based on the frequency of each label in data

  - Given all cars, how much more common are minivans than sports cars?

  - If we have no prior knowledge, we may assume that both classes are equally likely, i.e., $P(C_0)/P(C_1) = 1$

- $P(x \mid C_0)$ & $P(x \mid C_1)$: Comes from estimates of distributions

  - Estimate the distributions of the two classes given the labeled data!

$$P(x \mid C_0) \frac{P(C_0)}{P(C_1)} \overset{?}{>} P(x \mid C_1)$$
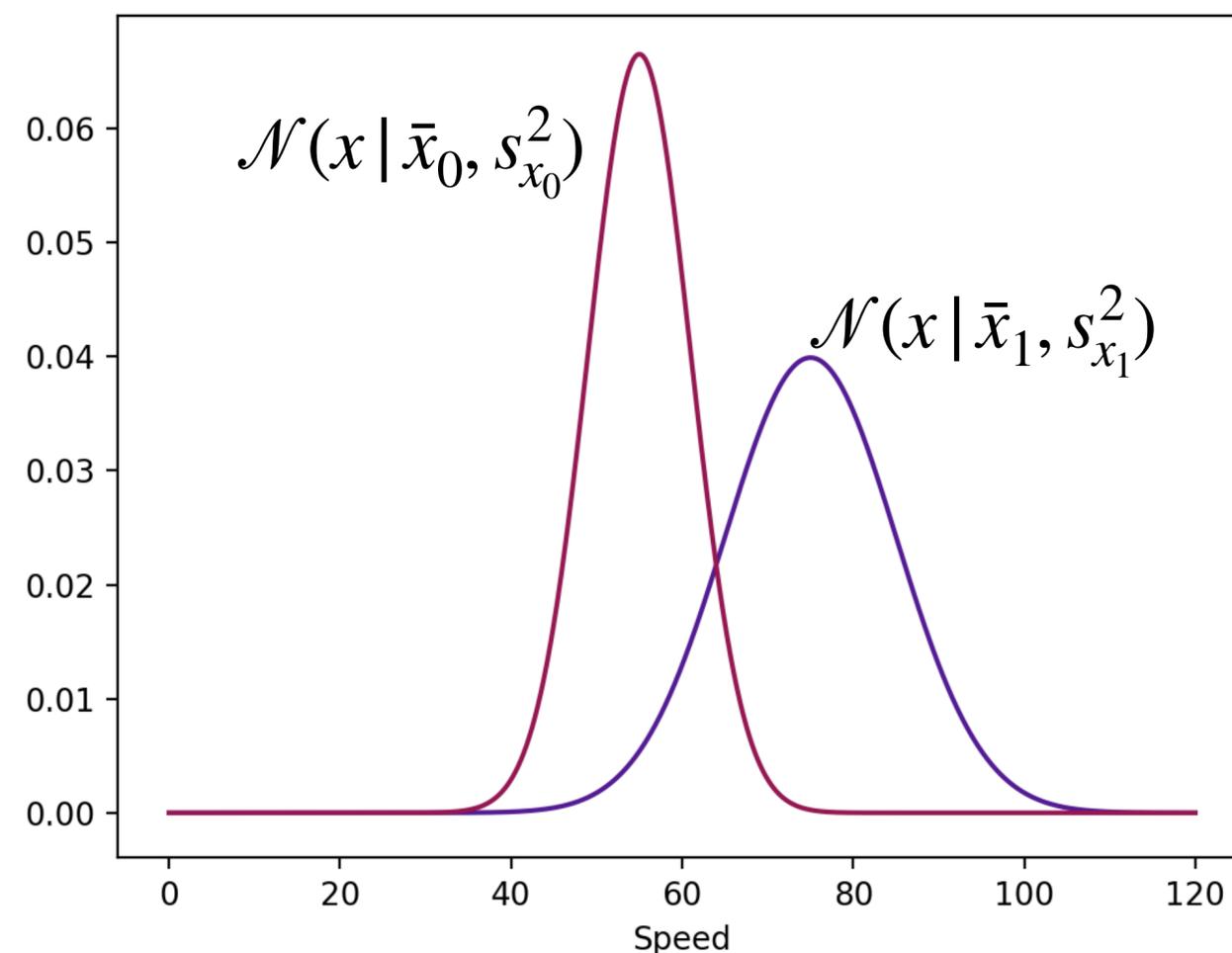

Naive Bayes Model

estimating distributions

- Can always fall back on empirical distributions

  - Use datasets to build histograms, use histograms as estimators (recall the histogram module)

- But the parametric approach is more common with Naïve Bayes

  - Use some prior knowledge to choose a model for your data, estimate the parameters of that model

- Common choice: **Gaussian Naïve Bayes**

  - Estimate mean and variance from data sets

  - Given a learned Gaussian, can directly "read off" the likelihood of x, i.e., $\mathcal{N}(x \mid \mu, \sigma^2)$

check the likelihood

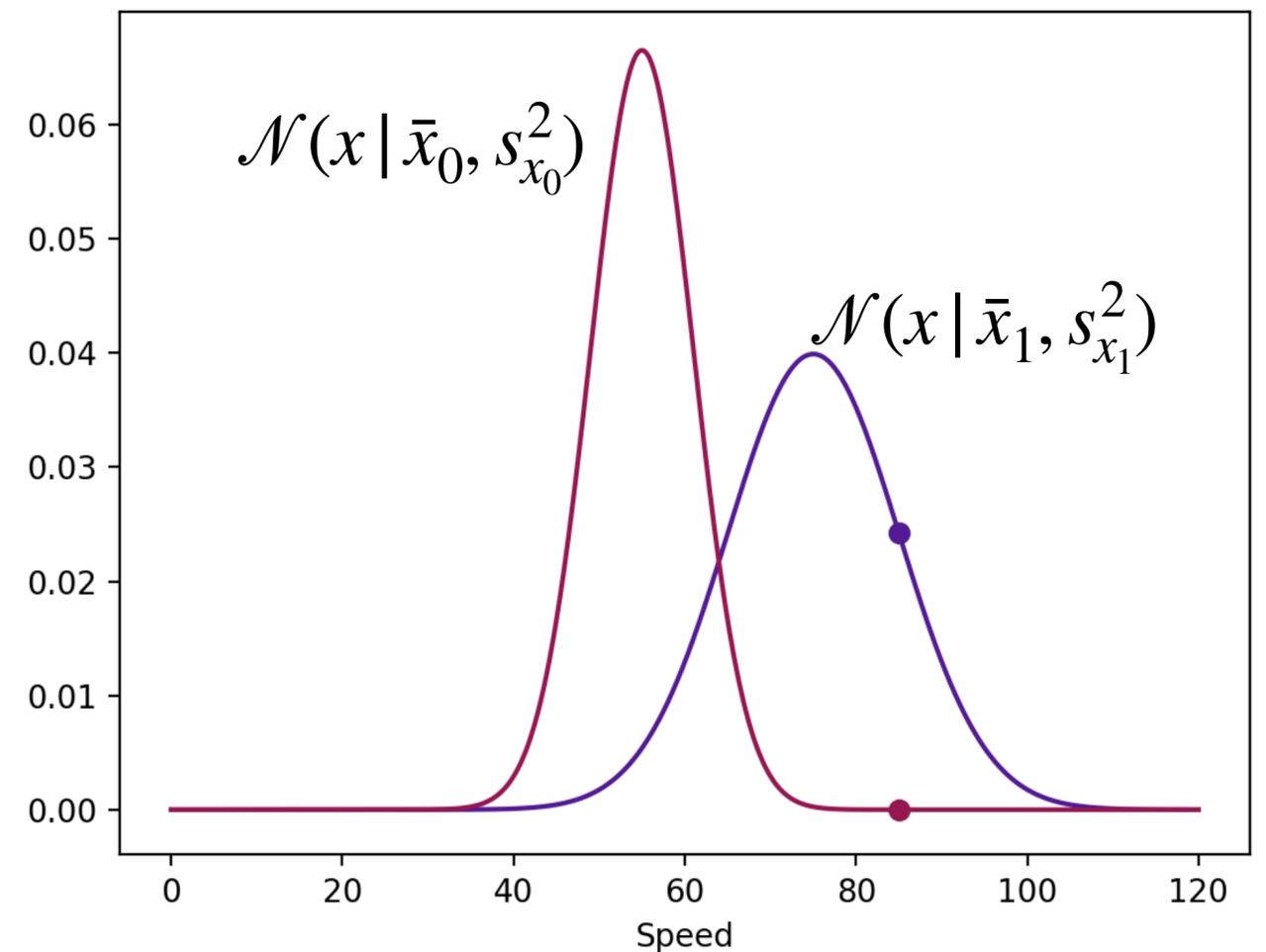$\mathcal{N}(\mu_1, \sigma_1^2)$ $\mathcal{N}(\mu_2, \sigma_2^2)$

back to our example: naïve bayes with gaussian

- Estimate parameters of **sports cars** and **minivans** by using mean and variance of training data

  - For all sports cars (C_1) in the training data, find the sample mean ($\bar{x}_1$) and sample variance ($s_{x_1}^2$), and use these to approximate μ_1 and σ_1^2

  - Likewise for all minivans (μ_0 and σ_0^2)

- Use resulting normal distributions to compute posterior probabilities of new data point being in one class or the other based on observed speed

naïve bayes with gaussian

- Estimate parameters of **sports cars** and **minivans** by using mean and variance of training data

- Use resulting normal distributions to compute likelihoods of new data point being in one class or the other based on observed speed

- If speed is high, more likely to be **sports car**

$\mathcal{N}(x \mid \bar{x}_0, s_{x_0}^2)$

$\mathcal{N}(x \mid \bar{x}_1, s_{x_1}^2)$

Speed

Note: here we are assuming that P(minivan) = P(sports car)
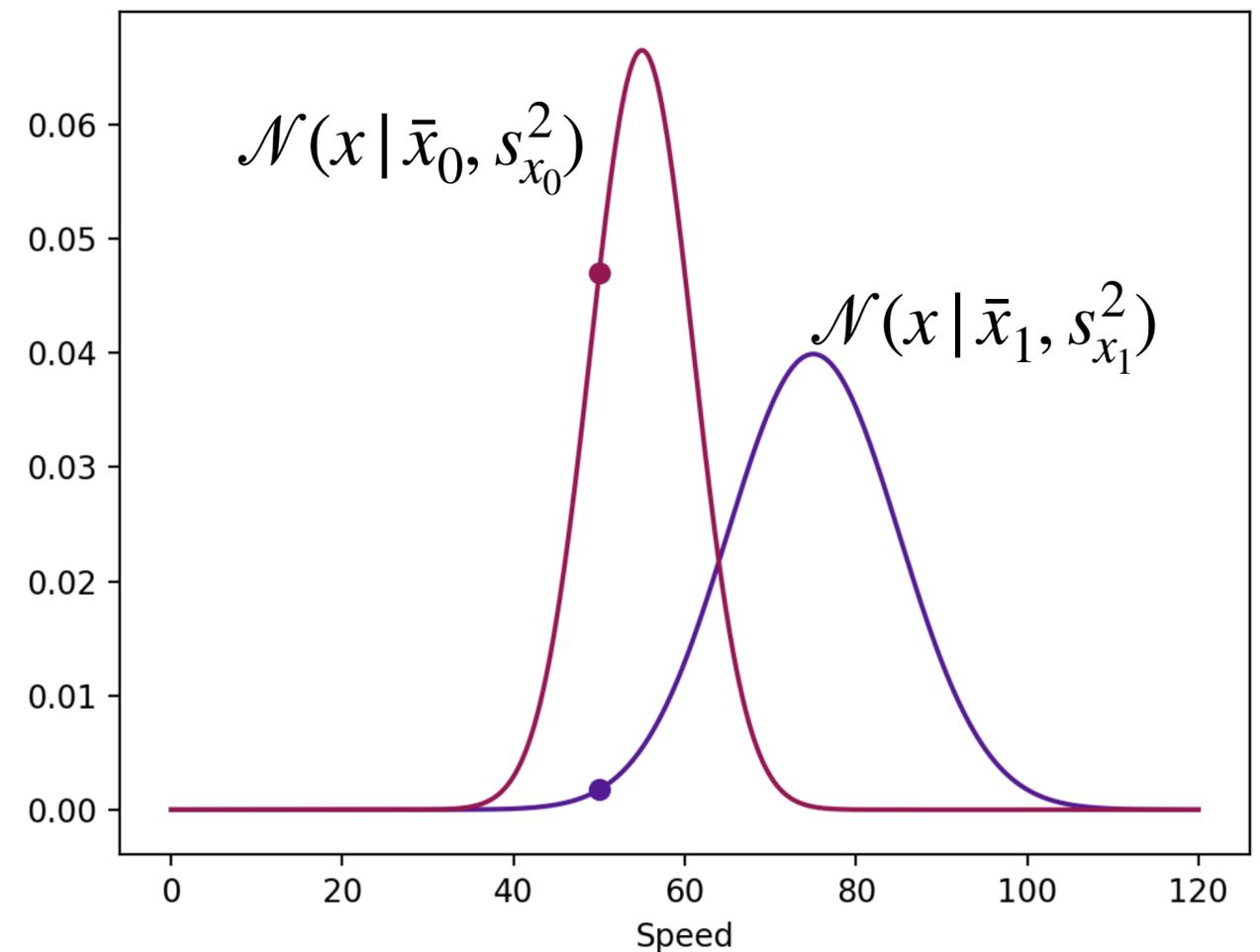
naïve bayes with gaussian

- Estimate parameters of **sports cars** and **minivans** by using mean and variance of training data

- Use resulting normal distributions to compute likelihoods of new data point being in one class or the other based on observed speed

- If speed is low, more likely to be **minivan**

Note: here we are assuming that P(minivan) = P(sports car)
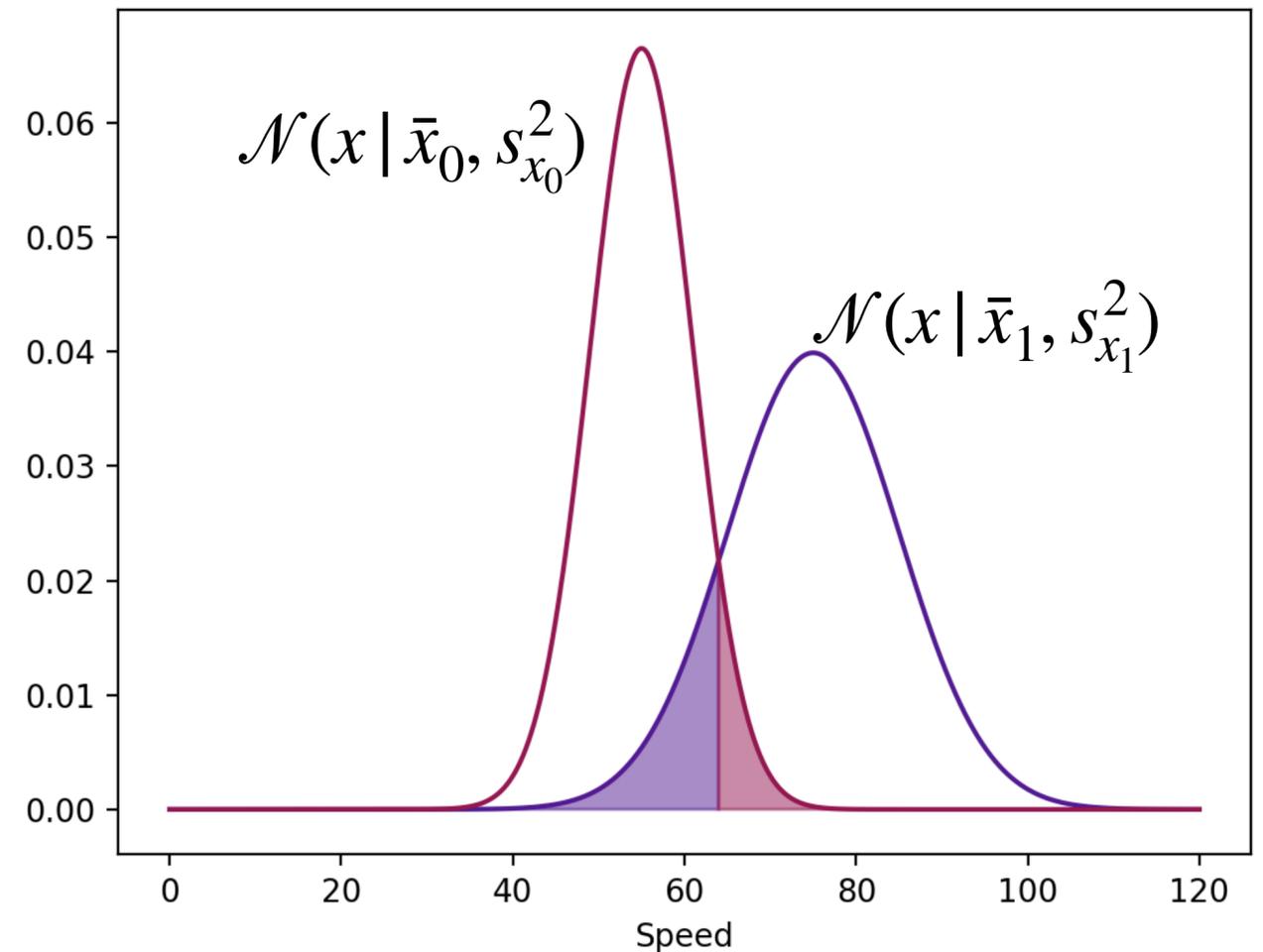
naïve bayes with gaussian

- Estimate parameters of **sports cars** and **minivans** by using mean and variance of training data

- Use resulting normal distributions to compute likelihoods of new data point being in one class or the other based on observed speed

- Note that it is possible to misclassify!

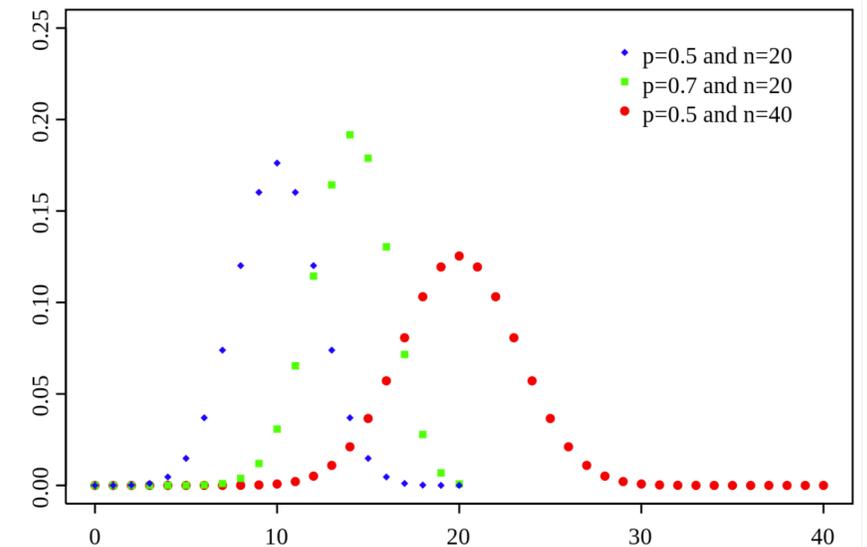**Note: here we are assuming that P(minivan) = P(sports car)**

what about higher dimensions?

Multinomial Distribution (2 count features)

- Suppose we have d features for each datapoint instead of just one

- For count features such as n-gram counts, could use **multinomial naïve Bayes**

- For continuous features, could use a **multivariate Gaussian**:

  - Naïve assumption: Assume features independent and fit class mean and variance for each feature *independently*

  - More general: Fit class in terms of mean (of each dimension) and **covariance matrix** (`cov` command in `numpy` computes covariance)

  - Mean is *d* x 1 vector μ, covariance is *d* x *d* matrix Σ with determinant $|\Sigma|$. The probability density function for a is:
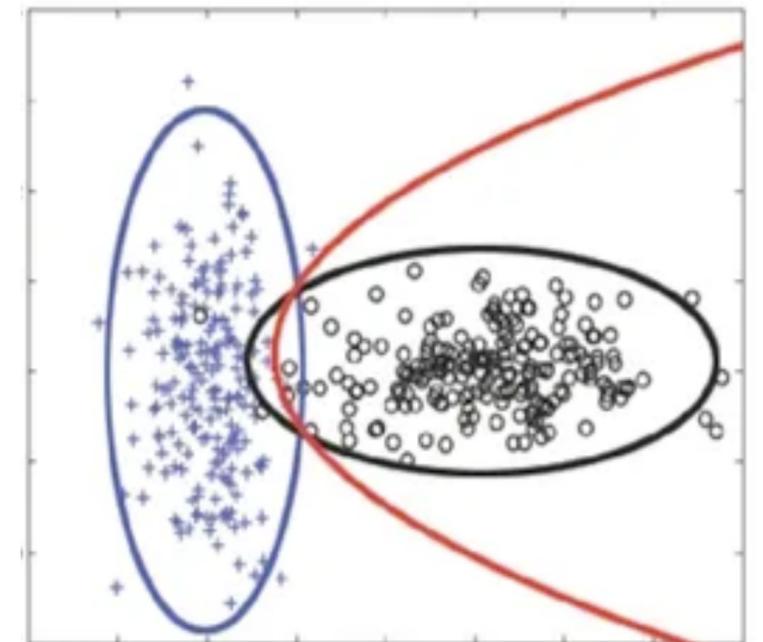
$$\mathcal{N}(\mathbf{x} \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d \mid \Sigma \mid}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)}$$

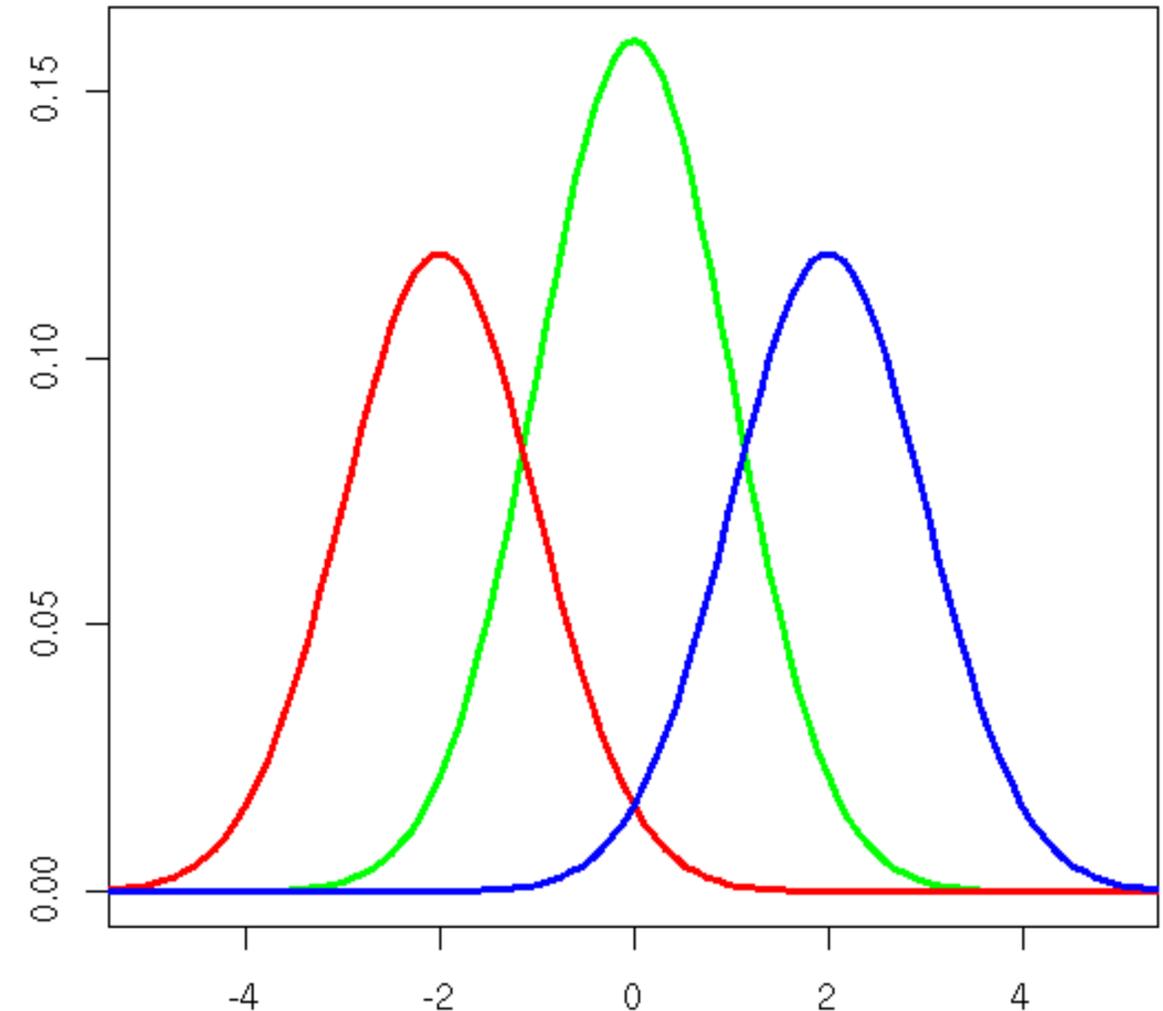**See: scipy.stats.multivariate_normal**

Gaussian Naive Bayes

what about multiple classes?

- We just infer and compare more cases

- For K classes, the predicted class for datapoint x is

$$\underset{k=0,...,K-1}{\arg\max} \ P(x \,|\, C_k)P(C_k)$$

- Issue with more classes is we have less data with which to infer each class' $P(x \,|\, C_k)$

  - The data is being "divided" further

naïve bayes in Python

- The `sklearn.naive_bayes` library (https://scikit-learn.org/stable/modules/classes.html#module-sklearn.naive_bayes)

- Three main types of interest

  - Gaussian (continuous data):
    `from sklearn.naïve_bayes import GaussianNB`

  - Bernoulli (binary data):
    `from sklearn.naïve_bayes import BernoulliNB`

  - Multinomial (count data):
    `from sklearn.naïve_bayes import MultinomialNB`

```python
from sklearn.datasets import load_iris
iris = load_iris()

X = iris.data
y = iris.target

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

y_pred = gnb.predict(X_test)

from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*100)
```

example of multinomial naïve bayes

We want to build a classifier to infer whether a new Tweet is about sports or not. Our training data has five tweets:

Text	Label
"A great game"	Sports
"The election was over"	Not Sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not Sports

Will the Tweet "A very close game" be assigned to Sports or Not Sports by a naïve bayes classifier?

To do this, assume a multinomial model where features are word frequencies (in a bag of words model). Do not filter out any stopwords.

solution

We have two classes, `Sports` and `Not Sports`, which we will call `S` and `N`. We need to compare the probabilities $P(S|\text{Tweet})$ and $P(N|\text{Tweet})$.

By the mechanics of naïve Bayes, we can reverse the conditional probabilities, so we need to compare

$$P(\text{Tweet}|S) \cdot P(S) \qquad \text{with} \qquad P(\text{Tweet}|N) \cdot P(N)$$

To compute $P(\text{Tweet}|\,\cdot\,)$, we are going to assume that every word in the sentence is independent of the others (similar to the bag of words principle). So,

$$P(\text{Tweet}|S) = P(\text{a}|S) \cdot P(\text{very}|S) \cdot P(\text{close}|S) \cdot P(\text{game}|S)$$

Multinomial distribution with N=4 words

$$P(\text{Tweet}|N) = P(\text{a}|N) \cdot P(\text{very}|N) \cdot P(\text{close}|N) \cdot P(\text{game}|N)$$

solution: probabilities and Laplace smoothing

Now, we infer $P(\text{word}|S)$ and $P(\text{word}|N)$ for each word by calculating the fraction of the class that corresponds to this particular word. For example, $P(\text{game}|S) = 2/11$, since out of the `sports` documents, `game` appears twice, and there are 11 words total.

Notice, however, that some of these probabilities would be 0, since not every word in the Tweet appears in both classes. For example, $P(\text{close}|S) = 0$. This would make $P(\text{Tweet}|S) = 0$ and $P(\text{Tweet}|N) = 0$, which is not particularly useful.

As a result, in solving these types of problems, we typically add something called **Laplace smoothing** to the probability calculations: we add 1 to each count in the numerator, and add the total number of words to the denominator to normalize.

solution

Text	Label
"A great game"	Sports
"The election was over"	Not Sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not Sports

Formally, the Laplace smoothing equations are:

$$P(\text{word} \,|\, S) = \frac{n_{w,S} + 1}{n_S + W} \quad \text{and} \quad P(\text{word} \,|\, N) = \frac{n_{w,N} + 1}{n_N + W}$$

where e.g., n_S is the number of words in the sports class and W is the total number of (unique) words.

In our case, $W = 14$. So, for example,
$P(\text{game} \,|\, S) = (2 + 1)/(11 + 14) = 3/25.$

The full set of probabilities are given in the table on the right.

word	P(word | S)	P(word | N)
a	3/25	2/23
very	2/25	1/23
close	1/25	2/23
game	3/25	1/23

solution

Text	Label
"A great game"	Sports
"The election was over"	Not Sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not Sports

word	P(word | S)	P(word | N)
a	3/25	2/23
very	2/25	1/23
close	1/25	2/23
game	3/25	1/23

Now, we multiply the probabilities:

$$P(\text{Tweet} \mid S) = \frac{3}{25} \cdot \frac{2}{25} \cdot \frac{1}{25} \cdot \frac{3}{25} = 2.76 \times 10^{-5}$$

$$P(\text{Tweet} \mid N) = \frac{2}{23} \cdot \frac{1}{23} \cdot \frac{2}{23} \cdot \frac{1}{23} \approx 0.572 \times 10^{-5}$$

Finally, we need $P(S)$ and $P(N)$. If the training dataset is large enough and chosen randomly, one possibility is to let these be the proportion of documents in each category. But with the absence of such information, it is safest to assume $P(S) = P(N) = 0.5$. Therefore,

$$P(S \mid \text{Tweet}) \propto P(\text{Tweet} \mid S)P(S) = 1.38 \times 10^{-5}$$

$$P(N \mid \text{Tweet}) \propto P(\text{Tweet} \mid N)P(N) = 0.286 \times 10^{-5}$$

Our classifier would predict `Sports`, as we would desire in this case!

pros vs cons of naïve bayes

+ Easy to build classifier (once you estimate the model)

+ Easy to compute likelihood

+ Good for missing data (distribution lets you estimate behaviors of data points you don't have)

− Need to choose a model for the data (get it wrong, classifier will not be reliable)

− Need prior knowledge to build classifier (relative likelihood of classes)