FDS22 Practical 2

DaST Team

Foundations of Data Science

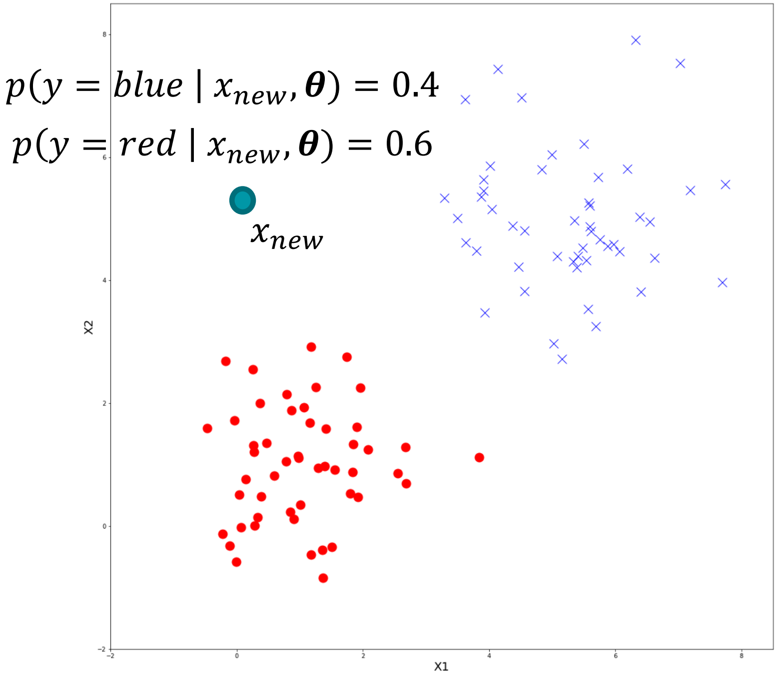
Outline

- Reproduce some of the experiment results in the paper
  On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes
  by Andrew Y. Ng and Michael I. Jordan

- Implement the generative classifier Naïve Bayes from scartch using NumPy

- Prepare the data for the experiments using Pandas

Discriminative Classifiers



Consider the dataset that has

- two features X1 and X2 and
- the labels with two classes {red, blue}

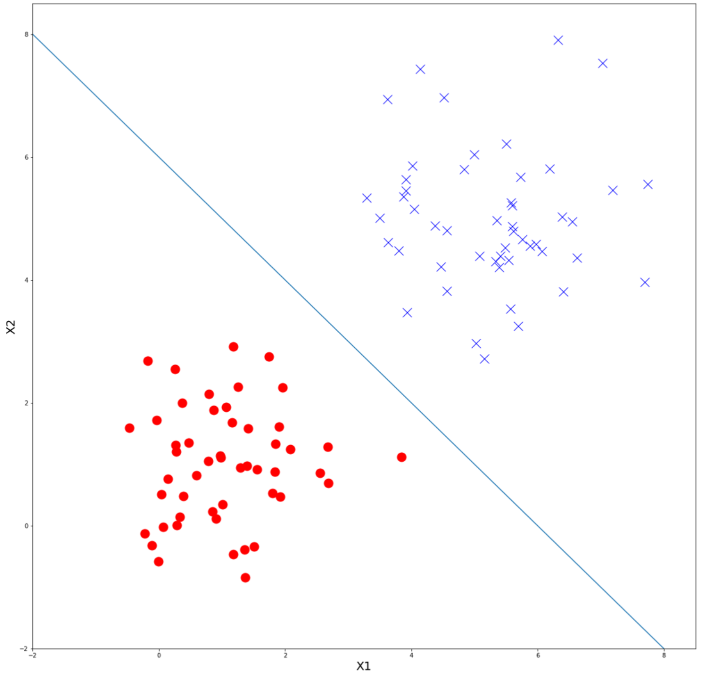
Discriminative classifiers:

$$p(y \mid \boldsymbol{x}, \boldsymbol{\theta})$$

Given a datapoint x, the probability that x belongs to a class y

e.g. $p(y = blue \mid \boldsymbol{x}, \boldsymbol{\theta})$ and $p(y = red \mid \boldsymbol{x}, \boldsymbol{\theta})$

Discriminative Classifiers



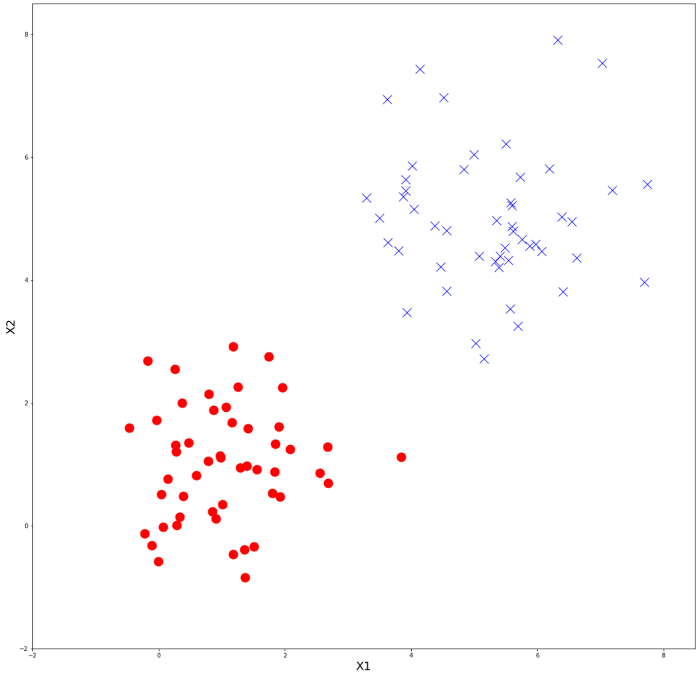
Discriminative classifiers learn the **decision boundary** that separates the data points with two classes.

Every datapoint on the decision boundary has the same probability of being red or blue:

$$p(y = blue \mid \boldsymbol{x}, \boldsymbol{\theta}) = p(y = red \mid \boldsymbol{x}, \boldsymbol{\theta})$$

Discriminative classifiers learn the “difference” between the two classes.

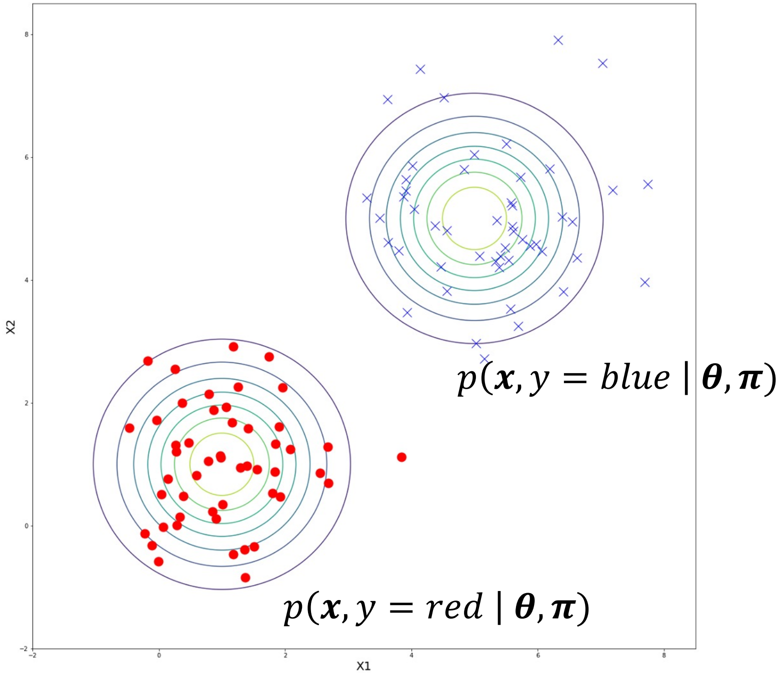
Generative Classifiers



Generative classifiers model the joint distribution:

$$p(\boldsymbol{x}, y \mid \boldsymbol{\theta}, \boldsymbol{\pi})$$

Generative Classifiers



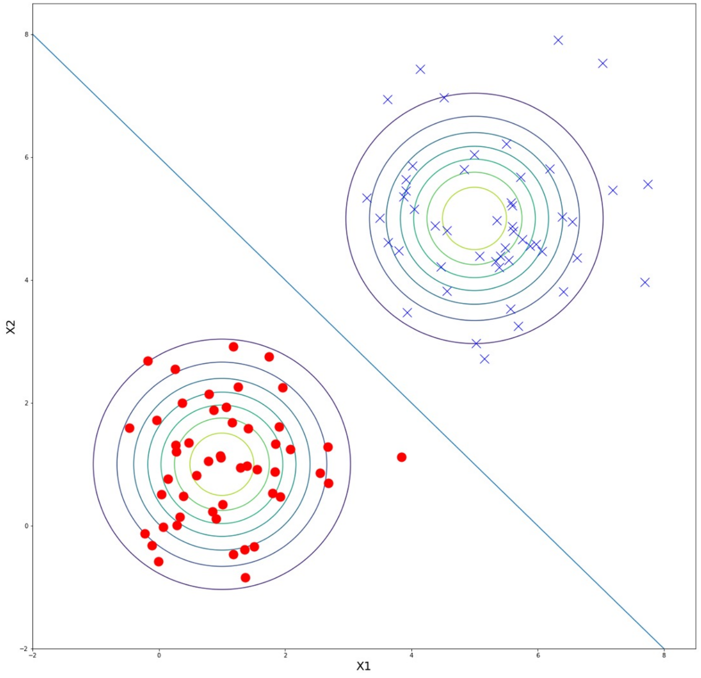
Generative classifiers model the joint distribution:

$$p(\boldsymbol{x}, y \mid \boldsymbol{\theta}, \boldsymbol{\pi})$$

Model the continuous data using multivariate Gaussian distributions

Generative classifiers learn how the data points in two classes look like.

Generative Classifiers



To classify the new data points, we derive the decision boundary from the joint distributions:

$$p(y = c \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = \frac{p(x_{new}, y = c \mid \boldsymbol{\theta}, \boldsymbol{\pi})}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})}$$

Classify the new data points as done in the discriminative case

Discriminative vs. Generative

Discriminative vs. Generative - Widely-held Beliefs

Two widely-held beliefs

- Asymptotic accuracy
    - Discriminative classifiers are almost always to be preferred to generative ones
    - “One should solve the [classification] problem directly and never solve a more general problem as an intermediate step [such as modelling $p(x, y|\theta, \pi)$]” by Vapnik
- Data efficiency
    - The number of records to fit a model is often roughly **linear** (or at most some low-order polynomial) in the number of parameters of a model

Discriminative vs. Generative - Widely-held Beliefs

Two widely-held beliefs

- Asymptotic accuracy
  - Discriminative classifiers are almost always to be preferred to generative ones
  - “One should solve the [classification] problem directly and never solve a more general problem as an intermediate step [such as modelling $p(x, y|\theta, \pi)$]” by Vapnik
- Data efficiency
  - The number of records to fit a model is often roughly **linear** (or at most some low-order polynomial) in the number of parameters of a model

Are these beliefs always true?

The paper studied the beliefs both empirically and theoretically.

Discriminative vs. Generative - Theoretical Conclusions

Generative classifiers:

- Stronger modelling assumptions, e.g.,
  - the choice of the distribution for modelling $p(\boldsymbol{x}, y \mid \boldsymbol{\theta}, \boldsymbol{\pi})$ and
  - features are conditionally independent given a class (used by naive bayes)
- Require less training data to learn “well”: O(logD)
  - if the assumptions are (approximately) correct
  - D is the number of parameters

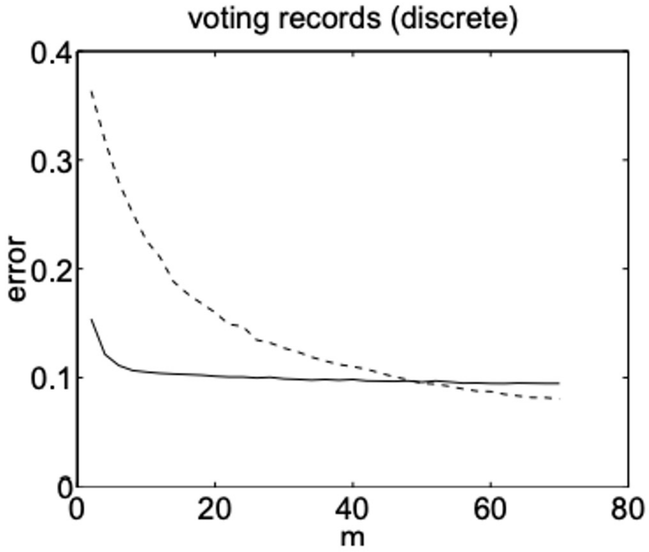
Discriminative classifiers:

- Significantly weaker assumptions (more **robust**)
- Require more training data: O(D)

Discriminative vs. Generative - Experiments

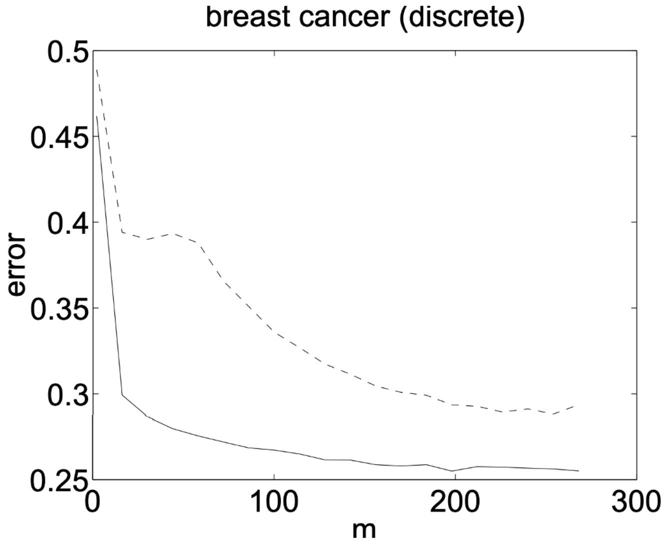
- Test logistic regression and naive bayes on 15 datasets
  - 8 with continuous inputs and
  - 7 with categorical inputs
- m: number of data points used for training
- Dashed line: logistic regression
- Solid line: naive bayes

- Even though the naive bayes classifier performs better initially, the logistic regression classifier eventually catches up and exceeds



Discriminative vs. Generative - Experiments

- Test logistic regression and naive bayes on 15 datasets
  - 8 with continuous inputs and
  - 7 with categorical inputs
- m: number of data points used for training
- Dashed line: logistic regression
- Solid line: naive bayes



- Logistic regression’s performance did not catch up to that of naive bayes
- This is observed primarily for small datasets for which m does not grow large enough

Read more: Murphy 8.6

Implement the Naïve Bayes

Implementation of a Naïve Bayes Classifier

- Implement a Naïve Bayes Classifier (NBC) as a class following the scikit-learn style

```
nbc = NBC() # initialise the model
nbc.fit(X_train, y_train) # train the model using the training data
y_pred = nbc.predict(X_test) # predict the label of the test data

accuracy = compute_accuracy(y_pred, y_test) # compare y_pred and y_test
```

Implementation of a Naïve Bayes Classifier

- Implement a Naïve Bayes Classifier (NBC) as a class following the scikit-learn style

```
nbc = NBC() # initialise the model
nbc.fit(X_train, y_train) # train the model using the training data
y_pred = nbc.predict(X_test) # predict the label of the test data

accuracy = compute_accuracy(y_pred, y_test) # compare y_pred and y_test
```

$p(\boldsymbol{x}, y \mid \boldsymbol{\theta}, \boldsymbol{\pi})$ joint distribution

- **fit** function: Estimates all parameters ($\boldsymbol{\theta}$ and $\boldsymbol{\pi}$) of the NBC using the training data
- **predict** function: Predicts the class of new input data

Predict the Class Label for a Given Input Data

- Assume we have estimated the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$
- Given a new input data x_{new}, predict the class label for x_{new}
- Compute for each class $c \in \{1, \dots, C\}$ a probability:

joint distribution

$$p(y = c \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = \frac{p(x_{new}, y = c \mid \boldsymbol{\theta}, \boldsymbol{\pi})}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})}$$

(by conditional probability formula)

The probability that x_{new} has class label c

Predict the Class Label for a Given Input Data

- Assume we have estimated the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$
- Given a new input data x_{new}, predict the class label for x_{new}
- Compute for each class $c \in \{1, \ldots, C\}$ a probability:

$p(y = 1 \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = 0.01$ The probability that x_{new} has the class label 1

$p(y = 2 \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = 0.25$ The probability that x_{new} has the class label 2

… …

$p(y = C \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = 0.03$ The probability that x_{new} has the class label C

- The prediction is the class that has the largest probability

$$y_{pred} = \underset{c}{argmax}\, p(y = c \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi})$$

Predict the Class Label for a Given Input Data

- Assume we have estimated the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$
- Given a new input data x_{new}, predict the class label for x_{new}
- Compute for each class $c \in \{1, \dots, C\}$ a probability:

joint distribution

$$p(y = c \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = \frac{p(x_{new}, y = c \mid \boldsymbol{\theta}, \boldsymbol{\pi})}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})}$$

(by conditional probability formula)

class prior

class-conditional distribution

$$= \frac{p(y = c \mid \boldsymbol{\pi}) \cdot p(x_{new} \mid y = c, \boldsymbol{\theta})}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})}$$

(by chain rule)

- The denominator is same for all classes, so we do not need to compute it

Class Prior $p(y = c \mid \boldsymbol{\pi})$

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyoncé
N	173K	CA	Beyoncé
Y	80K	NJ	Borat
Y	150K	WA	Beyoncé
N	25K	WV	Kanye West
Y	85K	IL	Beyoncé

- Class prior is only related to the labels, i.e., y-values
- In **fit** function:
  - $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_C\}$: Estimate a probability π_c for each class c
  - $\pi_c = p(y = c) = \frac{\#\, of\ appearance\ of\ class\ c}{\#\, of\ data}$
- Example:
  - $\boldsymbol{\pi} = \{\pi_{Beyonce}, \pi_{Borat}, \pi_{Kanye\ West}\}$
  - $\pi_{Beyonce} = p(y = Beyonce) = \frac{4}{6};\quad \pi_{Borat} = \frac{1}{6};\quad \pi_{Kanye\ West} = \frac{1}{6}$

Class Prior $p(y = c \mid \boldsymbol{\pi})$

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyoncé
N	173K	CA	Beyoncé
Y	80K	NJ	Borat
Y	150K	WA	Beyoncé
N	25K	WV	Kanye West
Y	85K	IL	Beyoncé

- Class prior is only related to the labels, i.e., y-values
- In **fit** function:
  - $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_C\}$: Estimate a probability π_c for each class c
  - $\pi_c = p(y = c) = \frac{\#\, of\ appearance\ of\ class\ c}{\#\, of\ data}$
- In **predict** function: For a class c
  - $p(y = c \mid \boldsymbol{\pi}) = \pi_c$
- Example:
  - $p(y = Beyonce \mid \boldsymbol{\pi}) = \pi_{Beyonce} = \frac{4}{6}$

Predict the Class Label for a Given Input Data

- Assume we have estimated the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\pi}$
- Given a new input data x_{new}, predict the class label for x_{new}
- Compute for each class $c \in \{1, \dots, C\}$ a probability:

joint distribution

$$p(y = c \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = \frac{p(x_{new}, y = c \mid \boldsymbol{\theta}, \boldsymbol{\pi})}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})}$$

(by conditional probability formula)

class prior

class-conditional distribution

$$= \frac{p(y = c \mid \boldsymbol{\pi}) \cdot p(x_{new} \mid y = c, \boldsymbol{\theta})}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})}$$

(by chain rule)

- The denominator is same for all classes, so we do not need to compute it

Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

$$p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$$

(by the conditional independence assumption of NB)

the probability for each feature j

The parameter $\boldsymbol{\theta}$ is a $j \times c$ matrix:

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{\boldsymbol{11}} & \cdots & \theta_{\boldsymbol{1c}} \\ \vdots & \ddots & \vdots \\ \theta_{\boldsymbol{jc}} & \cdots & \theta_{\boldsymbol{jc}} \end{pmatrix}$$

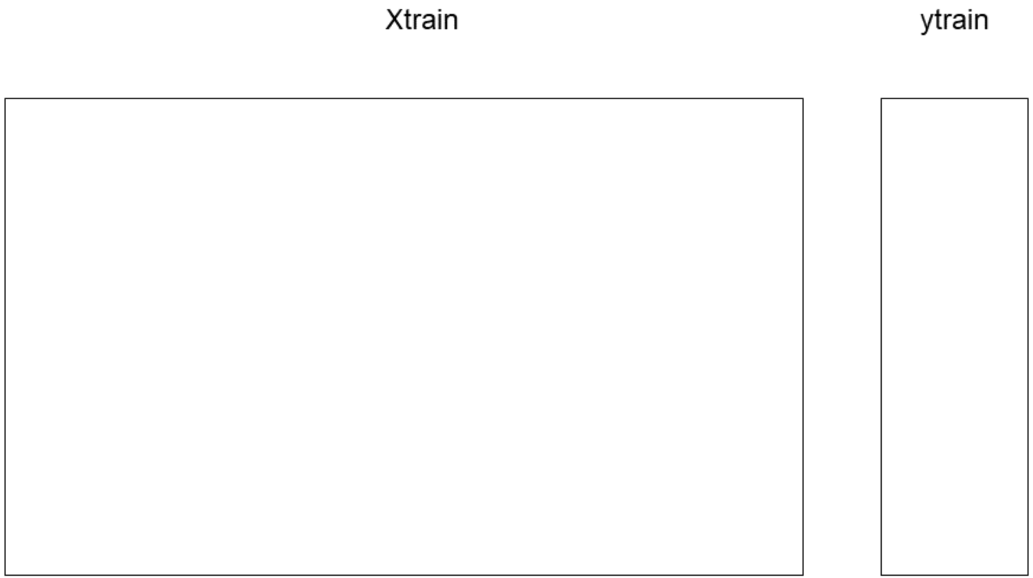
Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{\boldsymbol{11}} & \cdots & \theta_{\boldsymbol{1c}} \\ \vdots & \ddots & \vdots \\ \theta_{\boldsymbol{jc}} & \cdots & \theta_{\boldsymbol{jc}} \end{pmatrix}$$

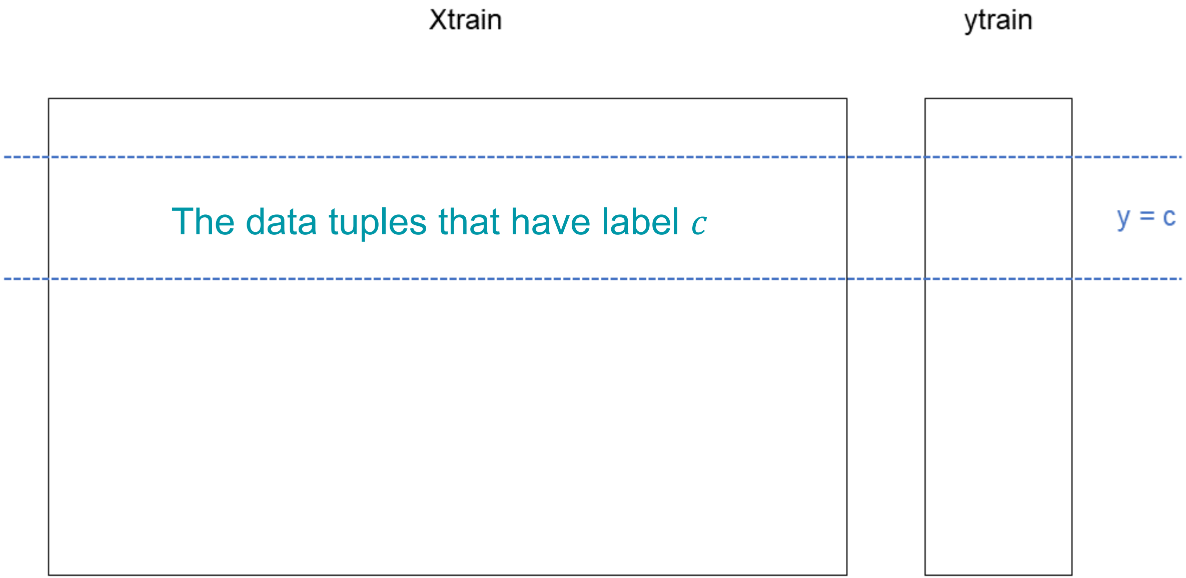
Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j



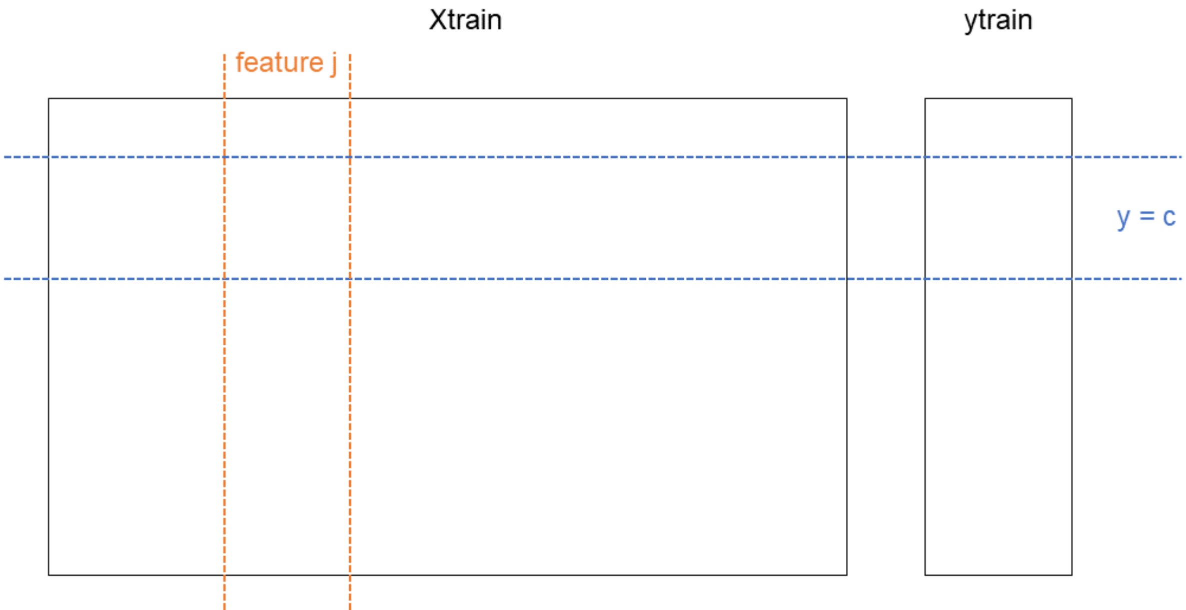
Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j



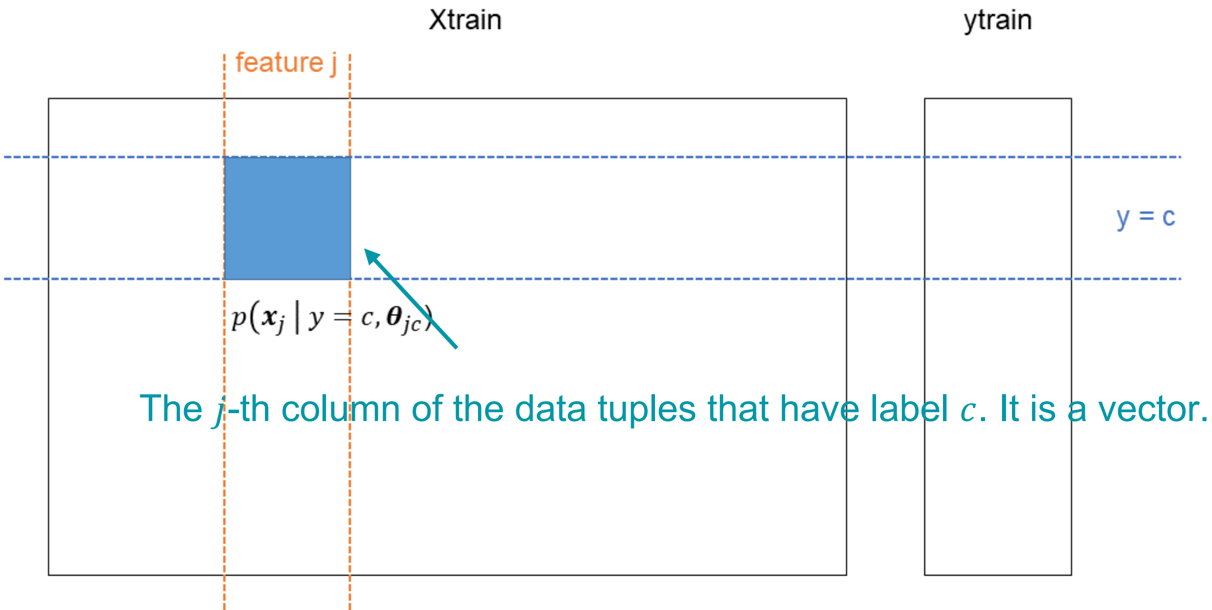
Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j



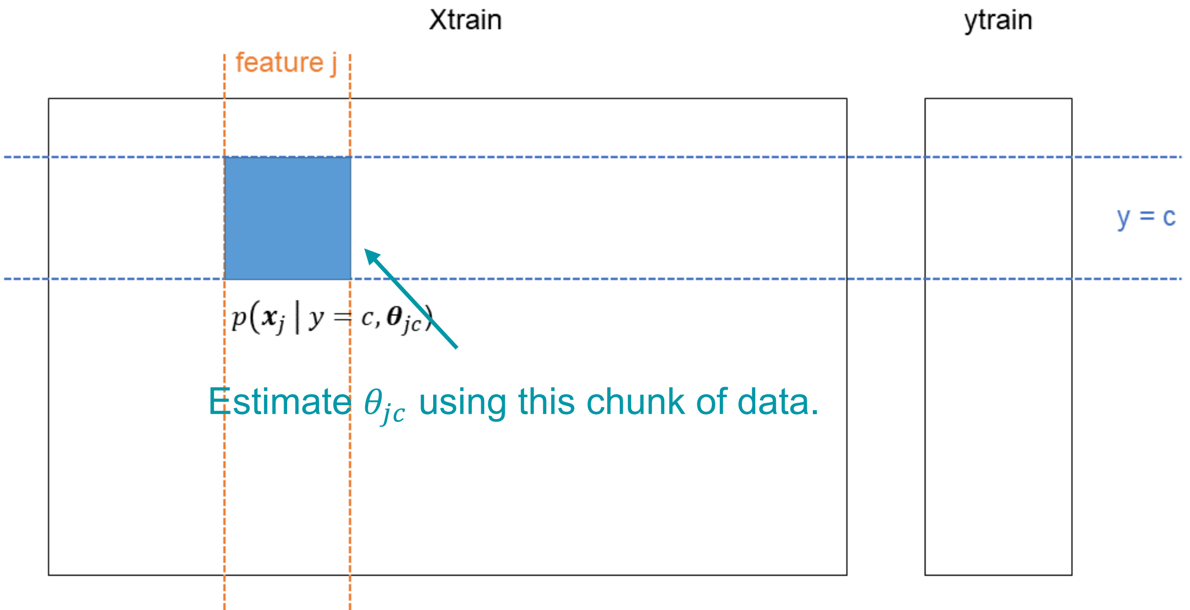
Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j



Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j



Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{Voted,Beyonce} & \theta_{Income,Beyonce} & \theta_{State,Beyonce} \\ \theta_{Voted,Borat} & \theta_{Income,Borat} & \theta_{State,Kanye} \\ \theta_{Voted,Kanye} & \theta_{Income,Kanye} & \theta_{State,Kanye} \end{pmatrix}$$

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyonce
N	173K	CA	Beyonce
Y	80K	NJ	Borat
Y	150K	WA	Beyonce
N	25K	WV	Kanye West
Y	85K	IL	Beyonce

Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{Voted,Beyonce} & \theta_{Income,Beyonce} & \theta_{State,Beyonce} \\ \theta_{Voted,Borat} & \theta_{Income,Borat} & \theta_{State,Kanye} \\ \theta_{Voted,Kanye} & \theta_{Income,Kanye} & \theta_{State,Kanye} \end{pmatrix}$$

Estimate $\theta_{Voted,Beyonce}$ using the data {Y, N, Y, Y}

- Binary data
- Bernoulli distribution

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyonce
N	173K	CA	Beyonce
Y	80K	NJ	Borat
Y	150K	WA	Beyonce
N	25K	WV	Kanye West
Y	85K	IL	Beyonce

Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{Voted,Beyonce} & \theta_{Income,Beyonce} & \theta_{State,Beyonce} \\ \theta_{Voted,Borat} & \theta_{Income,Borat} & \theta_{State,Kanye} \\ \theta_{Voted,Kanye} & \theta_{Income,Kanye} & \theta_{State,Kanye} \end{pmatrix}$$

Estimate $\theta_{Income,Beyonce}$ using the data {50, 173, 150, 85}

- Continuous data
- Gaussian distribution

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyonce
N	173K	CA	Beyonce
Y	80K	NJ	Borat
Y	150K	WA	Beyonce
N	25K	WV	Kanye West
Y	85K	IL	Beyonce

Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{Voted,Beyonce} & \theta_{Income,Beyonce} & \theta_{State,Beyonce} \\ \theta_{Voted,Borat} & \theta_{Income,Borat} & \theta_{State,Kanye} \\ \theta_{Voted,Kanye} & \theta_{Income,Kanye} & \theta_{State,Kanye} \end{pmatrix}$$

Estimate $\theta_{State,Beyonce}$ using the data {OK, CA, WA, IL}

- Categorical data
- Multinoulli distribution

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyonce
N	173K	CA	Beyonce
Y	80K	NJ	Borat
Y	150K	WA	Beyonce
N	25K	WV	Kanye West
Y	85K	IL	Beyonce

Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- In **predict** function: For a new input data x_{new},

$$p(x_{new} \mid y = c, \boldsymbol{\theta}) = \prod_{j} p\left(x_{new}^{j} \mid y = c, \theta_{jc}\right)$$

Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- In **predict** function: For a new input data x_{new},

$$p(x_{new} \mid y = c, \boldsymbol{\theta}) = \prod_j p\left(x_{new}^j \mid y = c, \theta_{jc}\right)$$

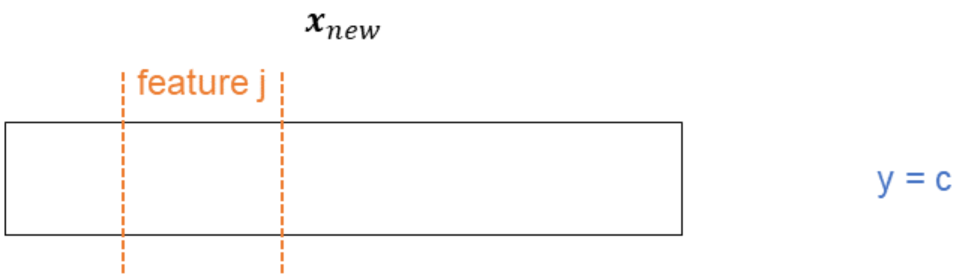
$\boldsymbol{x}_{new}$

y = c

Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- In **predict** function: For a new input data x_{new},

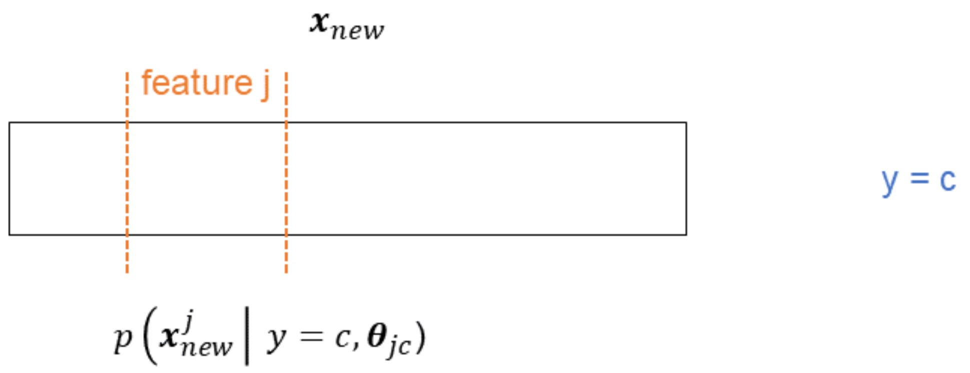
$$p(x_{new} \mid y = c, \boldsymbol{\theta}) = \prod_j p\left(x_{new}^j \mid y = c, \theta_{jc}\right)$$



Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- In **predict** function: For a new input data x_{new},

$$p(x_{new} \mid y = c, \boldsymbol{\theta}) = \prod_j p(x_{new}^j \mid y = c, \theta_{jc})$$



Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- In **predict** function: For a new input data x_{new},

$$p(x_{new} \mid y = c, \boldsymbol{\theta}) = \prod_j p\left(x_{new}^j \mid y = c, \theta_{jc}\right)$$

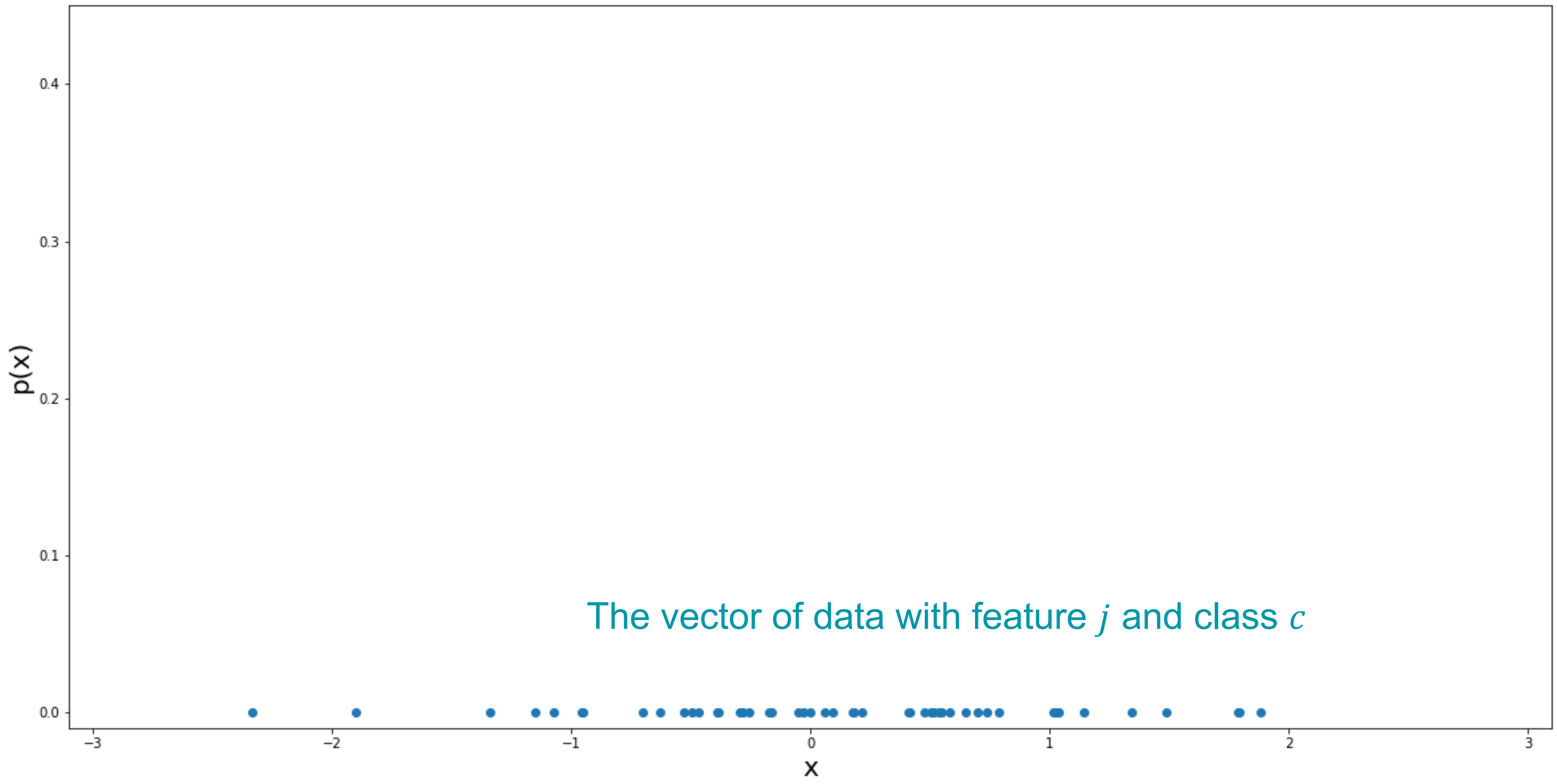
- For example, given the new input data $x_{new} = \{Y, 80K, WA\}$:

$$\begin{aligned} p(x_{new} \mid y = Beyonce, \boldsymbol{\theta}) = {} & p(Y \mid y = Beyonce, \theta_{Voted,\, Beyonce}) \cdot \\ & p(\, 80K \mid y = Beyonce, \theta_{Income,Beyonce}) \cdot \\ & p(\, WA \mid y = Beyonce, \theta_{State,Beyonce}) \end{aligned}$$

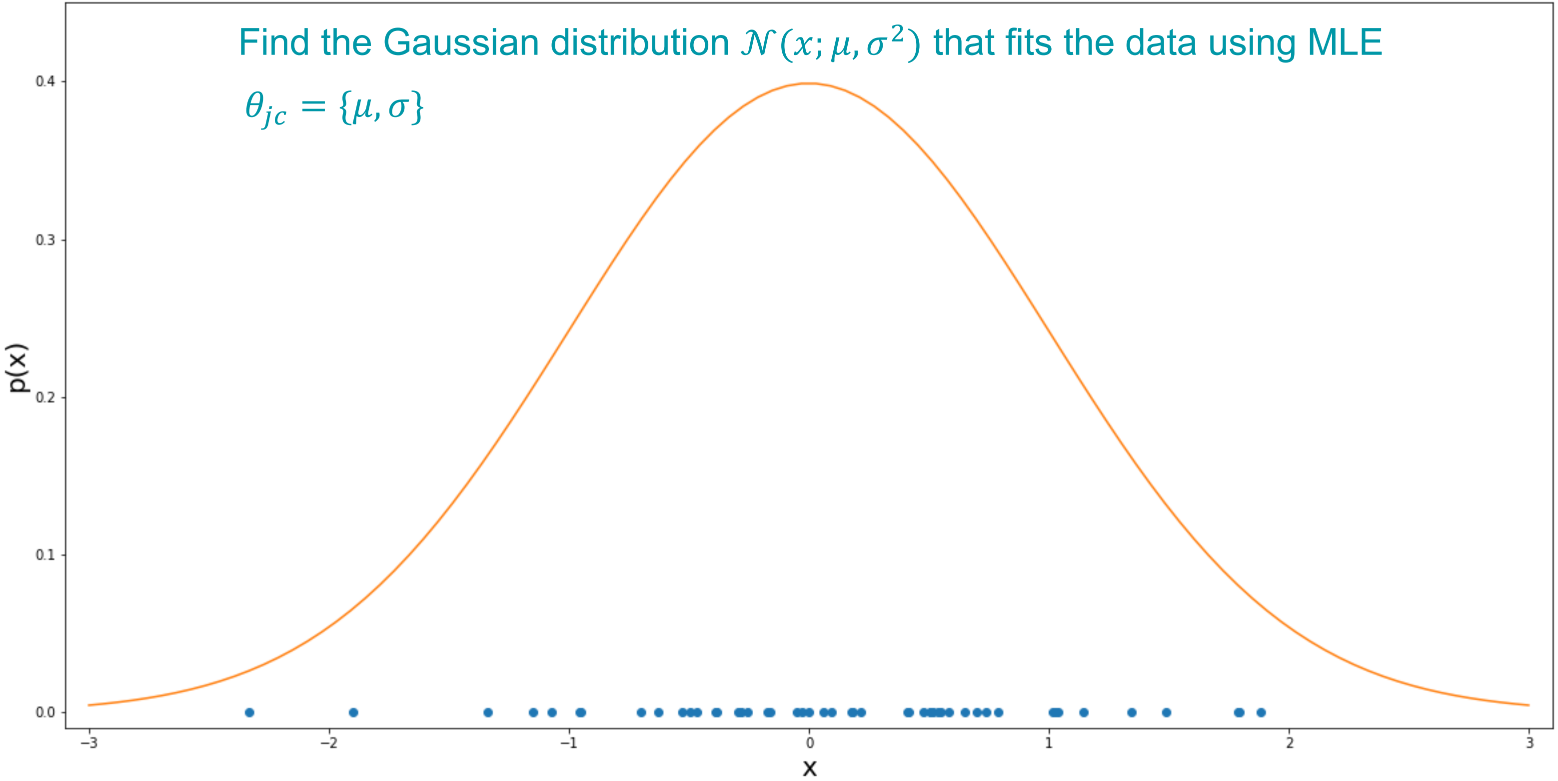
Estimation of θ_{jc}

- $p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$
- Use a distribution to model the data with feature j and class c
- θ_{jc} contains the parameters for the distribution
- The parameter θ_{jc} depends on the type of feature j
    - Continuous: Gaussian Distribution
    - Binary: Bernoulli Distribution
    - Categorical: Multinoulli Distribution

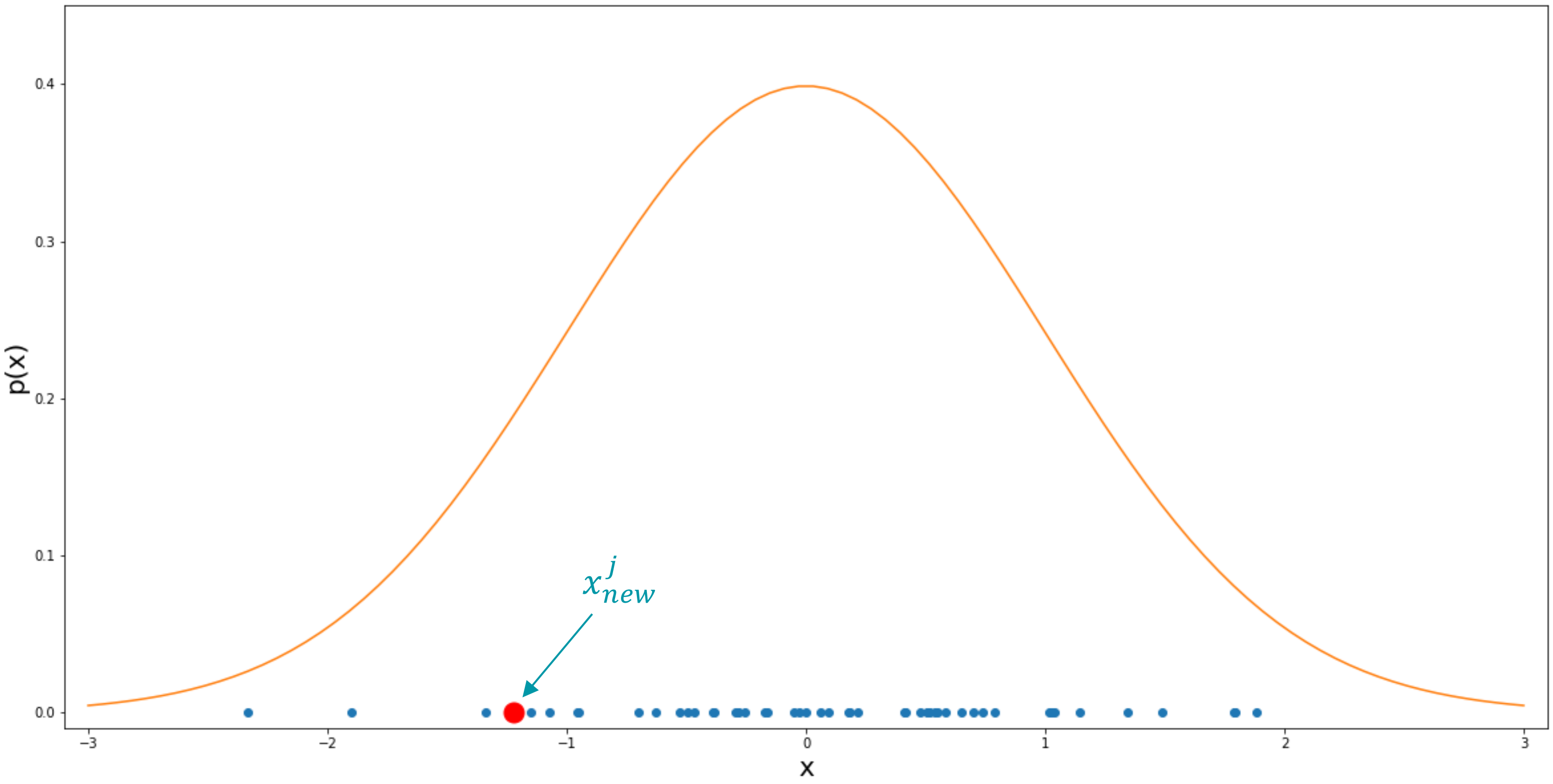
Estimation of θ_{jc}: Gaussian



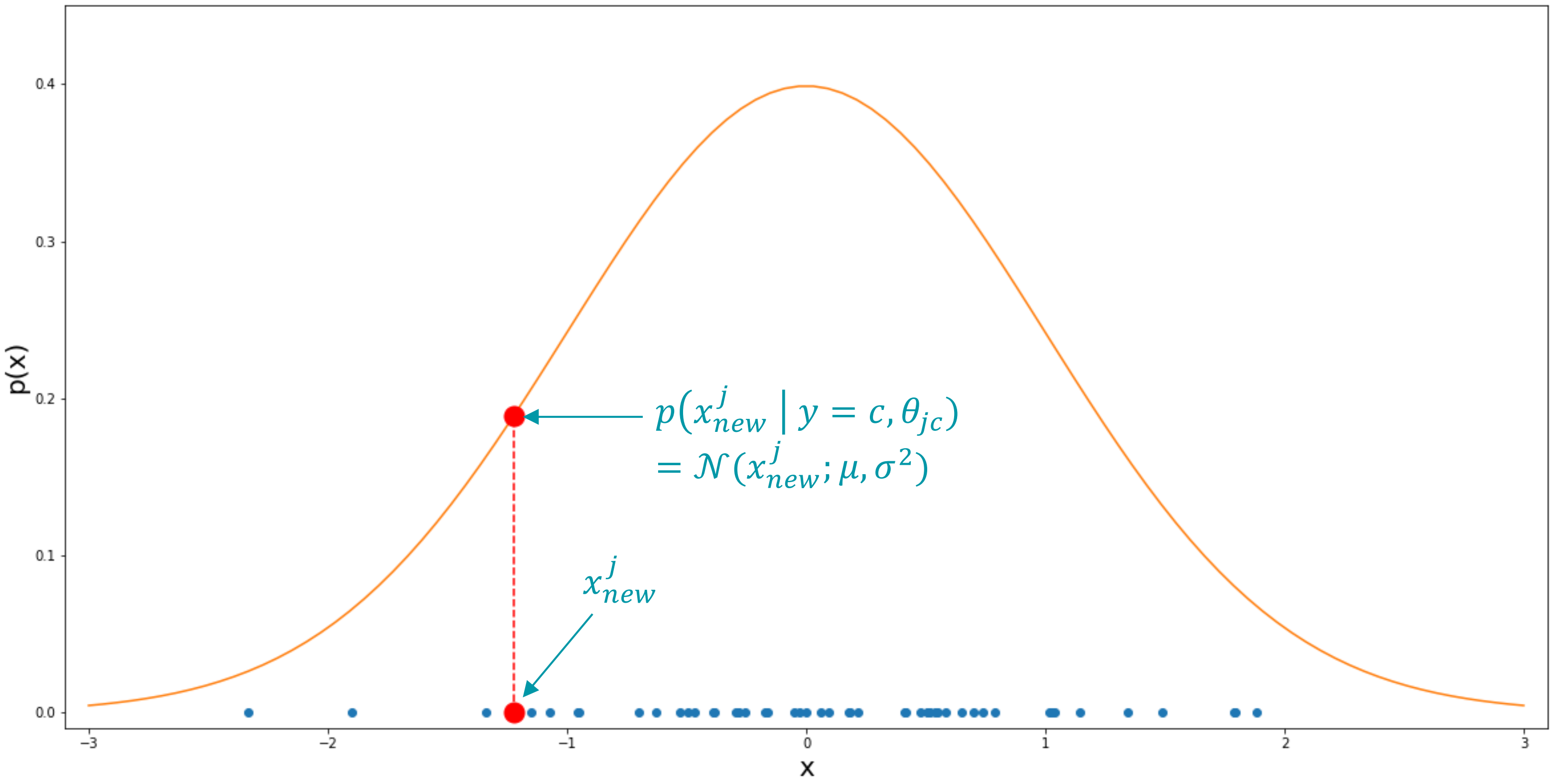
Estimation of θ_{jc}: Gaussian



Compute $p(x^j_{new} \mid y = c, \theta_{jc})$ using θ_{jc}: Gaussian



Compute $p(x_{new}^{j} \mid y = c, \theta_{jc})$ using θ_{jc}: Gaussian



Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j

$$\boldsymbol{\theta} = \left(\begin{array}{ccc} \theta_{Voted,Beyonce} & \theta_{Income,Beyonce} & \theta_{State,Beyonce} \\ \theta_{Voted,Borat} & \theta_{Income,Borat} & \theta_{State,Kanye} \\ \theta_{Voted,Kanye} & \theta_{Income,Kanye} & \theta_{State,Kanye} \end{array} \right)$$

Estimate $\theta_{Income,Beyonce}$ using the data {50, 173, 150, 85}

$$\theta_{Income,Beyonce} = \{\mu, \sigma\} = \{114.5, 49.278\}$$

For a new input datapoint with $x_{new}^{Income} = 100K$:

$$p(100 \mid y = Beyonce, \theta_{Income,Beyonce}) = \mathcal{N}(100; \mu, \sigma^2)$$

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyonce
N	173K	CA	Beyonce
Y	80K	NJ	Borat
Y	150K	WA	Beyonce
N	25K	WV	Kanye West
Y	85K	IL	Beyonce

Bernoulli Distribution

- The probability mass function is

$$f(x) = \begin{cases} p(x=0), & if\ x = 0 \\ p(x=1), & if\ x = 1 \end{cases}$$

where $x \in \{0, 1\}$ and $p(x = i) = \frac{\#\ of\ apperance\ of\ i}{\#\ of\ data}$

- Estimation: $\theta_{jc} = p(x = 1)$
- Prediction: $p(1\ |\theta_{jc}) = \theta_{jc}$ and $p(0\ |\theta_{jc}) = 1 - \theta_{jc}$

Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{Voted,Beyonce} & \theta_{Income,Beyonce} & \theta_{State,Beyonce} \\ \theta_{Voted,Borat} & \theta_{Income,Borat} & \theta_{State,Kanye} \\ \theta_{Voted,Kanye} & \theta_{Income,Kanye} & \theta_{State,Kanye} \end{pmatrix}$$

Estimate $\theta_{Voted,Beyonce}$ using the data {Y, N, Y, Y}

$$\theta_{Voted,Beyonce} = p(x = Y) = \frac{3}{4}$$

For a new input datapoint with $x_{new}^{Voted} = N$:

$$p(N \mid y = Beyonce, \theta_{Voted,Beyonce}) = 1 - \theta_{Voted,Beyonce} = \frac{1}{4}$$

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyonce
N	173K	CA	Beyonce
Y	80K	NJ	Borat
Y	150K	WA	Beyonce
N	25K	WV	Kanye West
Y	85K	IL	Beyonce

Multinoulli Distribution

- Optional task (bonus points)
- Also called Categorical distribution
- https://en.wikipedia.org/wiki/Categorical_distribution

- The Bernoulli distribution is a special case of the Multinoulli distribution

Multinoulli Distribution

- k distinct values in j-th column of the data
- The probability mass function is

$$f(x) = \begin{cases} p(x=1), & if\ x=1 \\ p(x=2), & if\ x=2 \\ & \dots \\ p(x=k), & if\ x=k \end{cases}$$

where $x \in \{1, \dots, k\}$ and $p(x=i) = \frac{\#\ of\ apperance\ of\ i}{\#\ of\ data}$

- Estimation: $\theta_{jc} = \{p(x=1), \dots, p(x=k)\}$
- Prediction: $p(i\ |\theta_{jc}) = p(x=i)$

Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$ (by the conditional independence assumption of NB)
- In **fit** function: Estimate a parameter θ_{jc} for each class c and each feature j

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_{Voted,Beyonce} & \theta_{Income,Beyonce} & \theta_{State,Beyonce} \\ \theta_{Voted,Borat} & \theta_{Income,Borat} & \theta_{State,Kanye} \\ \theta_{Voted,Kanye} & \theta_{Income,Kanye} & \theta_{State,Kanye} \end{pmatrix}$$

Voted in 2016?	Annual Income	State	Candidate Choice
Y	50K	OK	Beyonce
N	173K	CA	Beyonce
Y	80K	NJ	Borat
Y	150K	WA	Beyonce
N	25K	WV	Kanye West
Y	85K	IL	Beyonce

Estimate $\theta_{State,Beyonce}$ using the data {OK, CA, WA, IL}

$$\theta_{State,Beyonce} = \{p(x = OK), p(x = CA), p(x = NJ), \dots\} = \{\frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4}, 0, \frac{1}{4}\}$$

For a new input datapoint with $x_{new}^{State} = OK$:

$$p(OK \mid y = Beyonce, \theta_{State,Beyonce}) = \frac{1}{4}$$

Put Everything Together

- **fit** function: Estimates all parameters
  - $\boldsymbol{\pi} = \{\pi_1, \dots, \pi_C\}$
  - $\boldsymbol{\theta} = \{\theta_{jc}$ | for each class c and each feature $j\}$
- **predict** function: For a new data x_{new}, computes for each class c

$$p(y = c \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = \frac{p(y = c \mid \boldsymbol{\pi}) \cdot p(x_{new} \mid y = c, \boldsymbol{\theta})}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})}$$

$$= \frac{\pi_c \cdot \prod_j p\left(x_{new}^j \mid y = c, \theta_{jc}\right)}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})}$$

  - Choose the class with largest probability (no need to compute the denominator as it is same for all classes)

Pseudocode: fit

```
function fit(X_train, y_train):

    for each class c:

        // estimate class prior
        pi_c <- p(y=c)

        for each feature j:

            // get the data with class c and feature j
            X_jc <- X_train[y_train==c, j]

            // estimate theta_jc
            // the estimation should be based on the type of j
            theta_jc <- estimate theta_jc on X_jc
```

Pseudocode: predict

```
function predict(x_new):

    for each class c:

        prob_c = pi_c

        for each feature j:

            x_new_j = x_new[:,j]
            prob_c *= p(x_new_j | theta_jc)

    return class c with the largest prob_c
```

$$p(y = c \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = \frac{\pi_c \cdot \prod_j p(x_{new}^j \mid y = c, \theta_{jc})}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})}$$

- In this pseudocode, the input of the predict function is a single data point.
- In the skeleton code, the input of the predict function is a matrix with multiple data points. The function should predict the labels for all data points.

Skeleton Code

Data Preparation

See the Jupyter notebook: prepare_data.ipynb

- Data Cleaning (handling missing values)
- Handling Text and Categorical Features
- Feature Scaling
- Get Test Data