FDS21 Practical 2

DaST Team

Foundations of Data Science

Outline

- Implement Naïve Bayes from scartch

- 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

- Prepare the data for the experiments

Implementation of a Naïve Bayes Classifier

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

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

- **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

$\ldots$ — $\ldots$

$p(y = C \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = 0.03$ — 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:

$$p(y = c \mid x_{new}, \boldsymbol{\theta}, \boldsymbol{\pi}) = \frac{\overbrace{p(x_{new}, y = c \mid \boldsymbol{\theta}, \boldsymbol{\pi})}^{\text{joint distribution}}}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})} \quad \text{(by conditional probability formula)}$$

$$= \frac{\overbrace{p(y = c \mid \boldsymbol{\pi})}^{\text{class prior}} \cdot \overbrace{p(x_{new} \mid y = c, \boldsymbol{\theta})}^{\text{class-conditional distribution}}}{p(x_{new} \mid \boldsymbol{\theta}, \boldsymbol{\pi})} \quad \text{(by chain rule)}$$

- 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
- 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
- In **fit** function:
  - $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_C\}$: 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 $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

$$p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_{j}^{D} p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$$ (by the conditional independence assumption of NB)

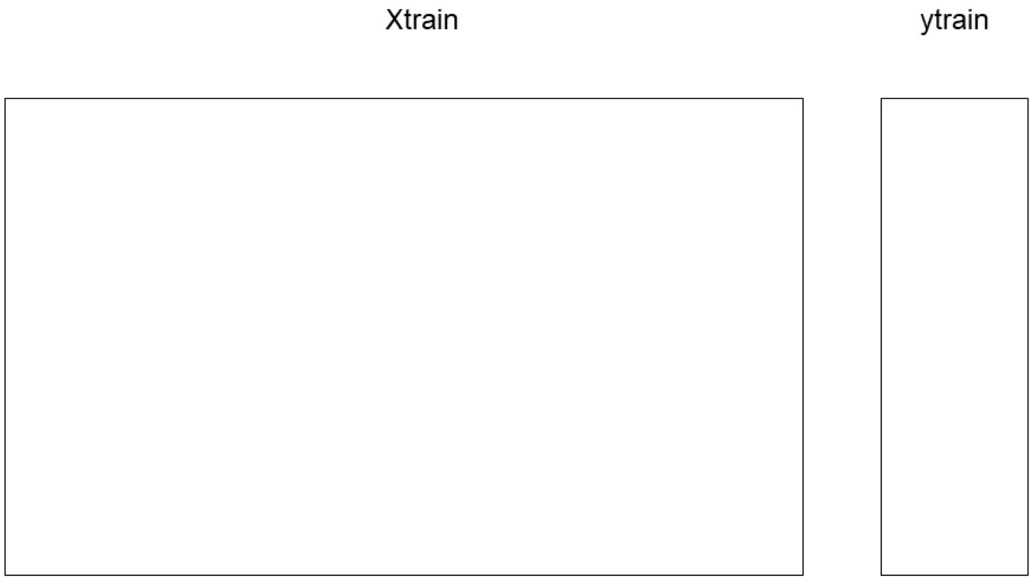
Compute the probability for each feature j

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

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

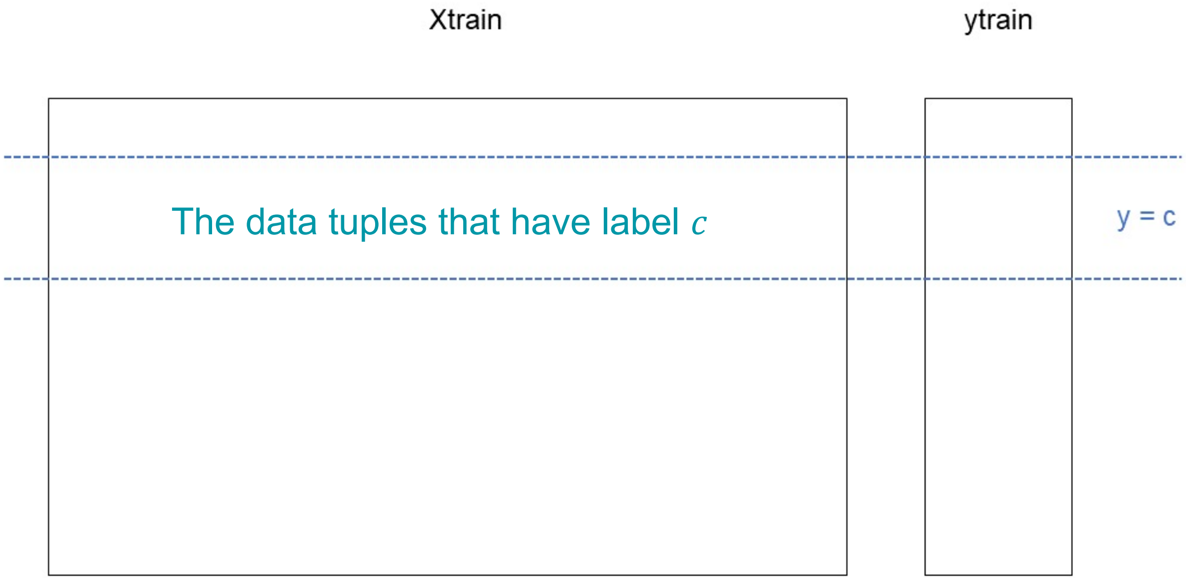
Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

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



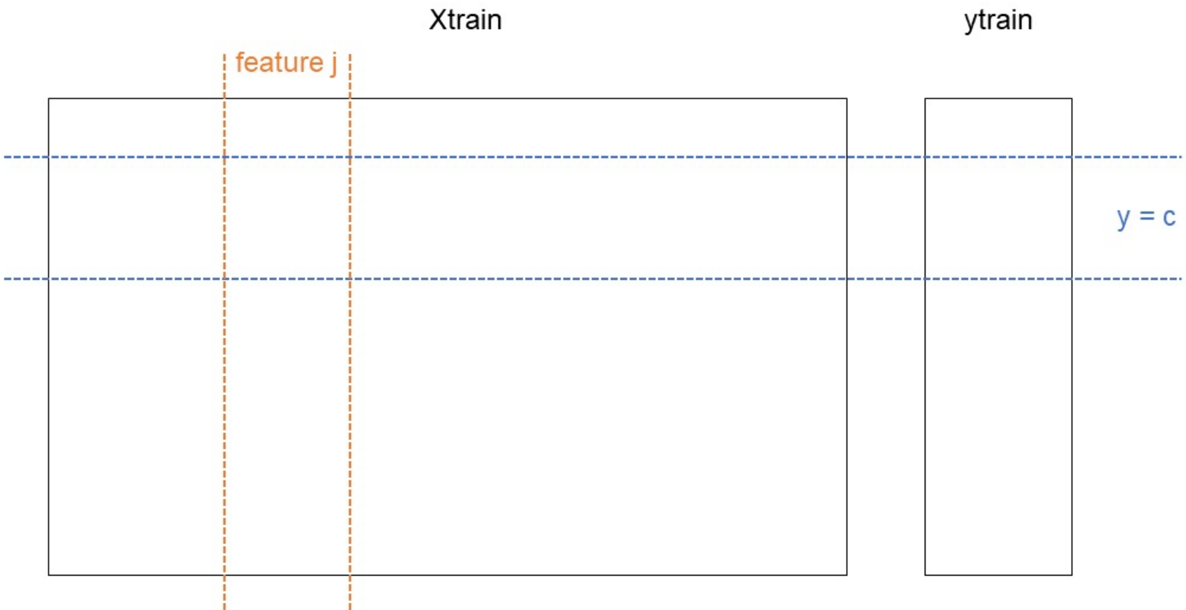
Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

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



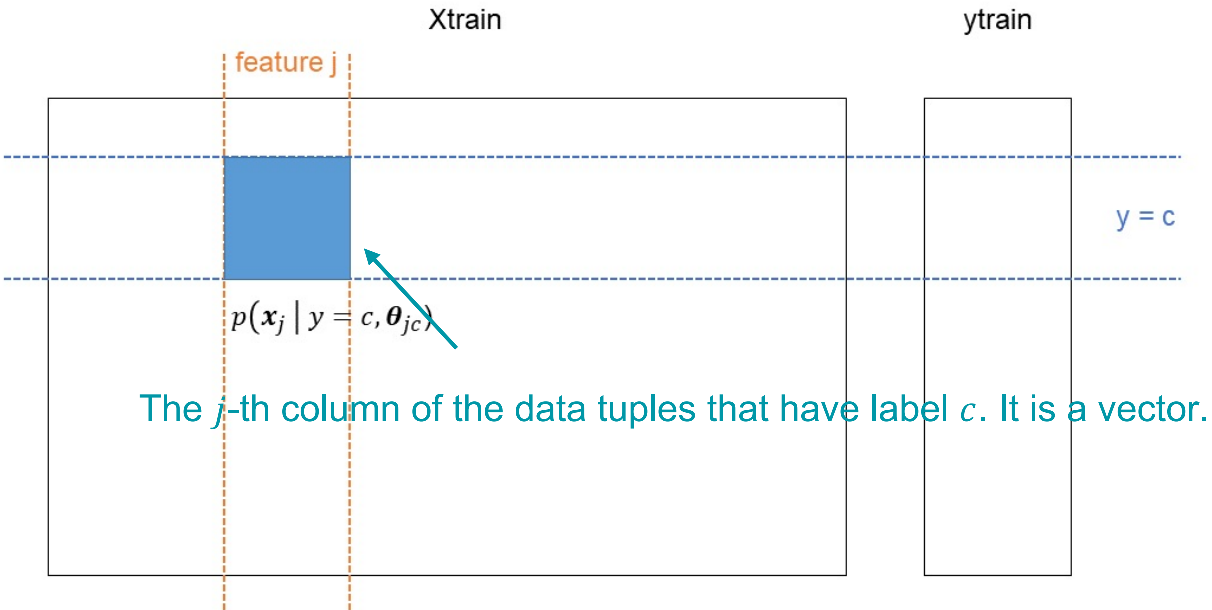
Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

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



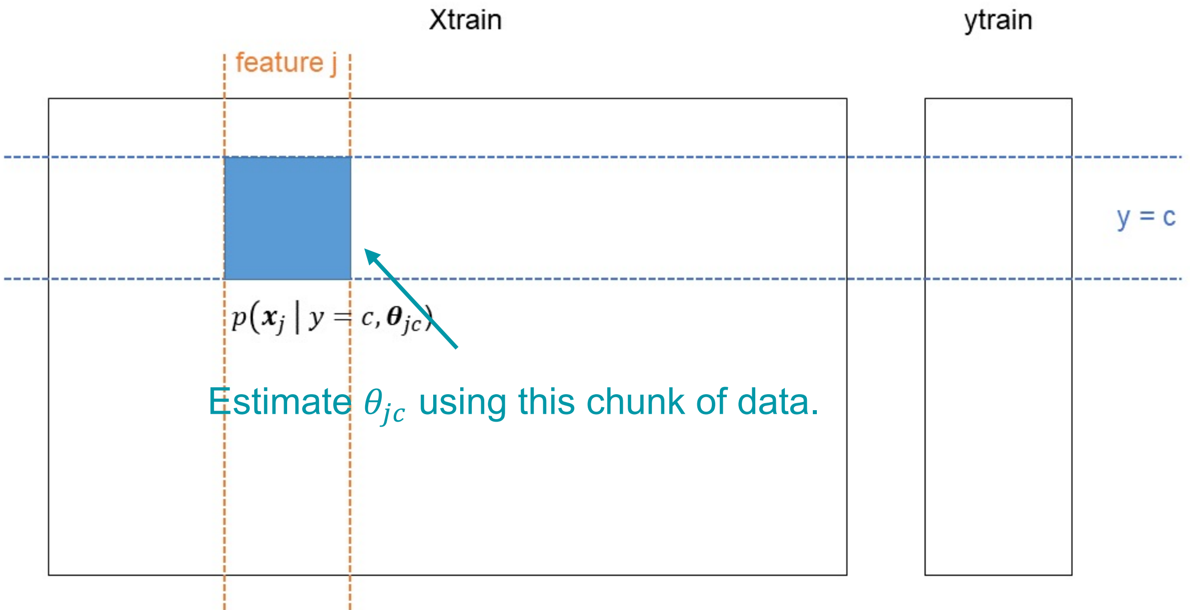
Class-conditional distribution $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

- $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_j^D p(\boldsymbol{x}_j \mid y = c, \theta_{jc})$ (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 $p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta})$

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

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

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

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

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

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

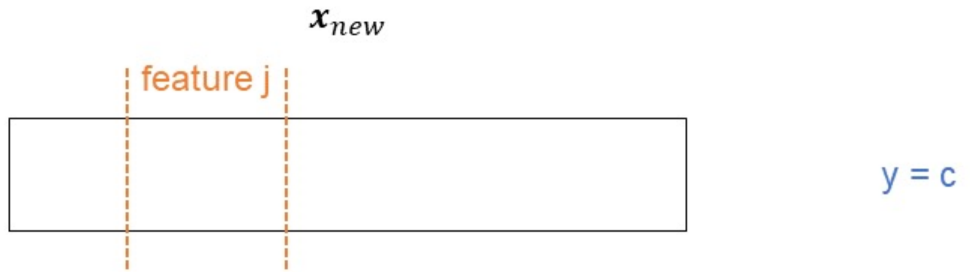
$\boldsymbol{x}_{new}$

y = c

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

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

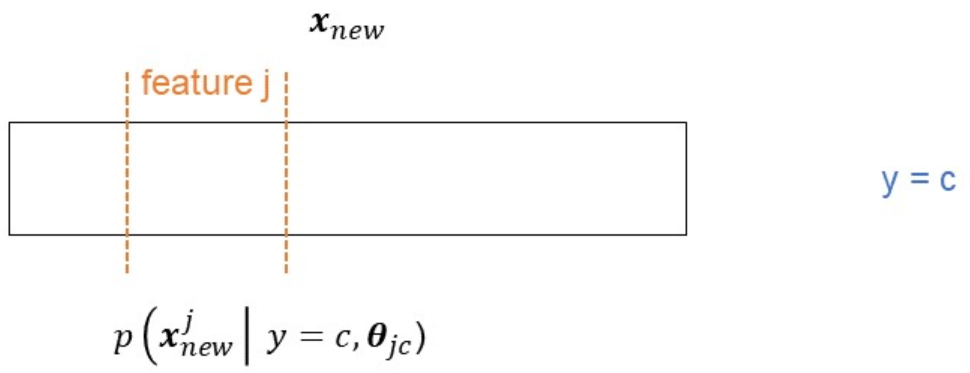
$$p(x_{new} \mid y = c, \boldsymbol{\theta}) = \prod_{j}^{D} p\left(x_{new}^{j} \mid y = c, \theta_{jc}\right)$$



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

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

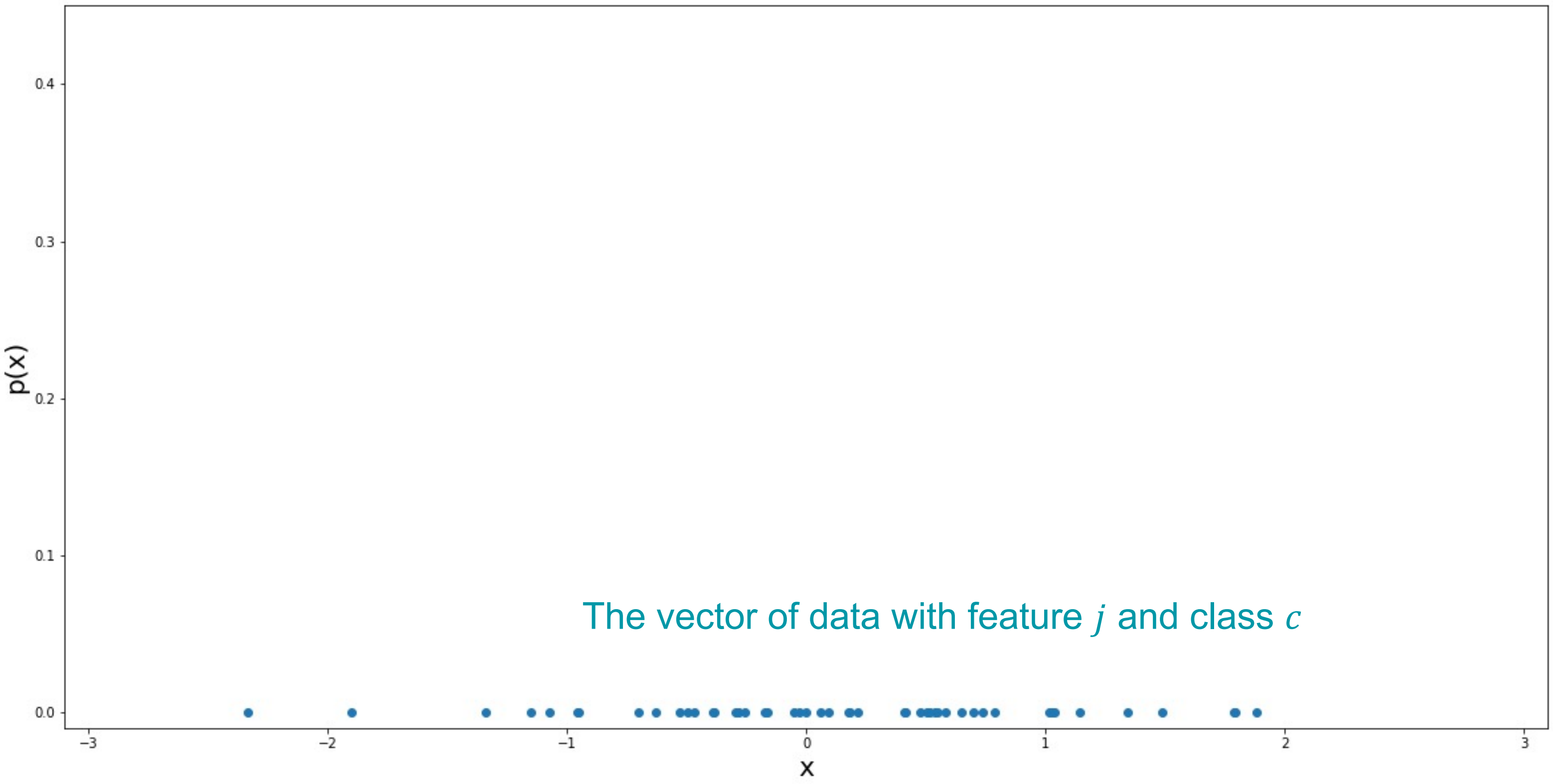
$$p(x_{new} \mid y = c, \boldsymbol{\theta}) = \prod_{j}^{D} p(x_{new}^{j} \mid y = c, \theta_{jc})$$



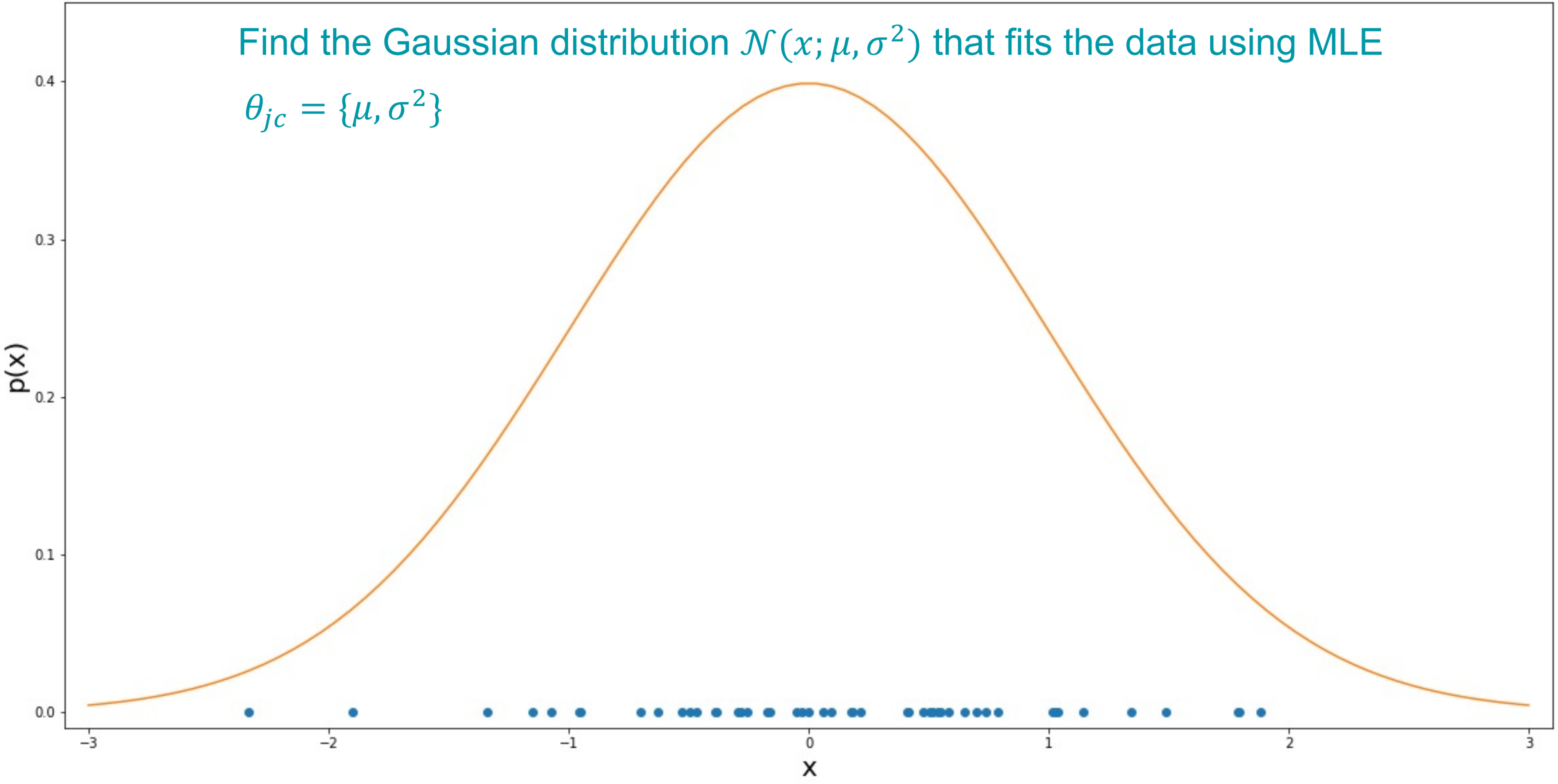
Estimation of θ_{jc}

- $p(\boldsymbol{x_j} \mid y = c, \theta_{jc})$
- Use a distribution to model the data with feature j and class c
- θ_{jc} represents 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

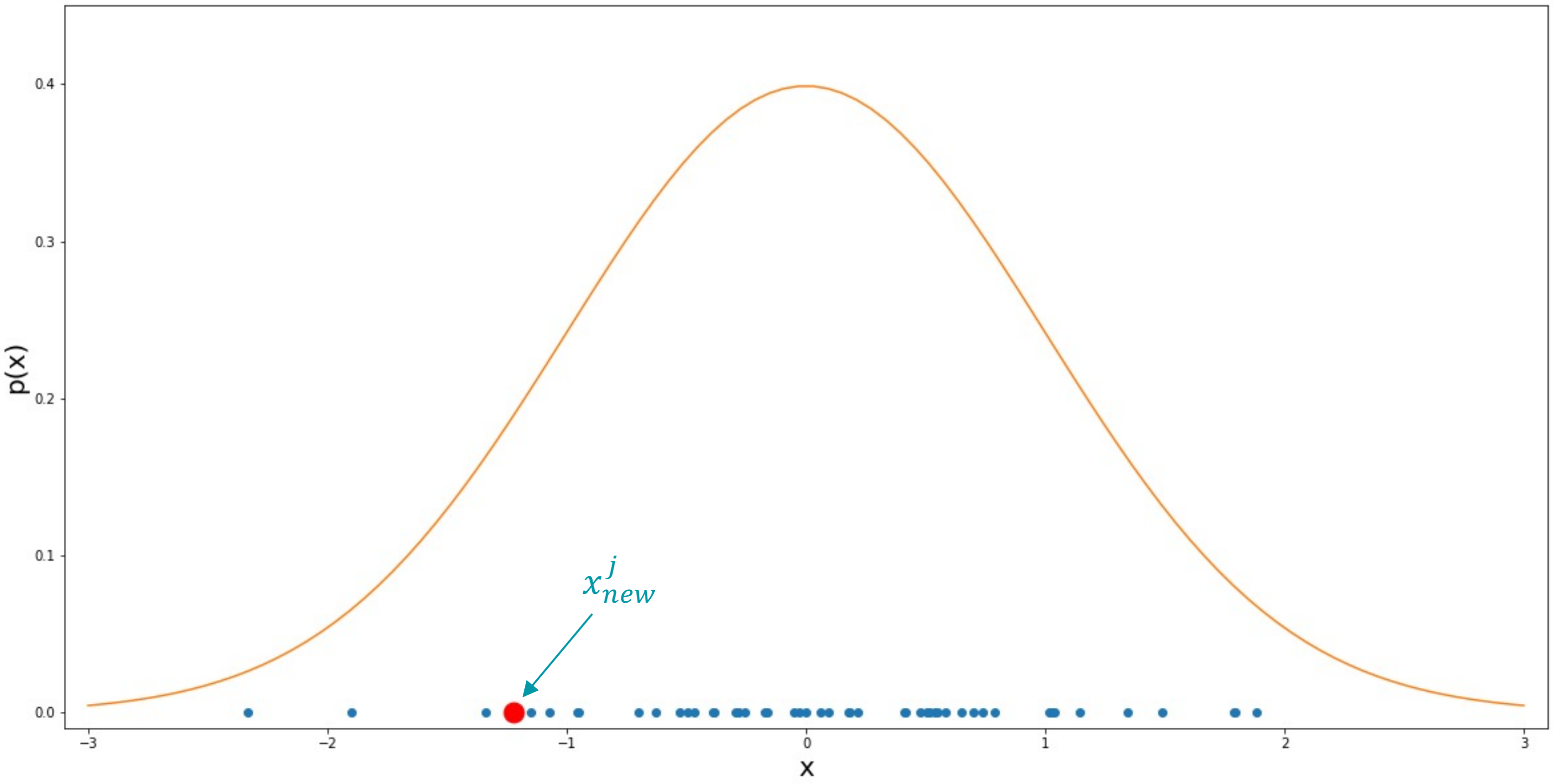
Estimation of θ_{jc}: Gaussian



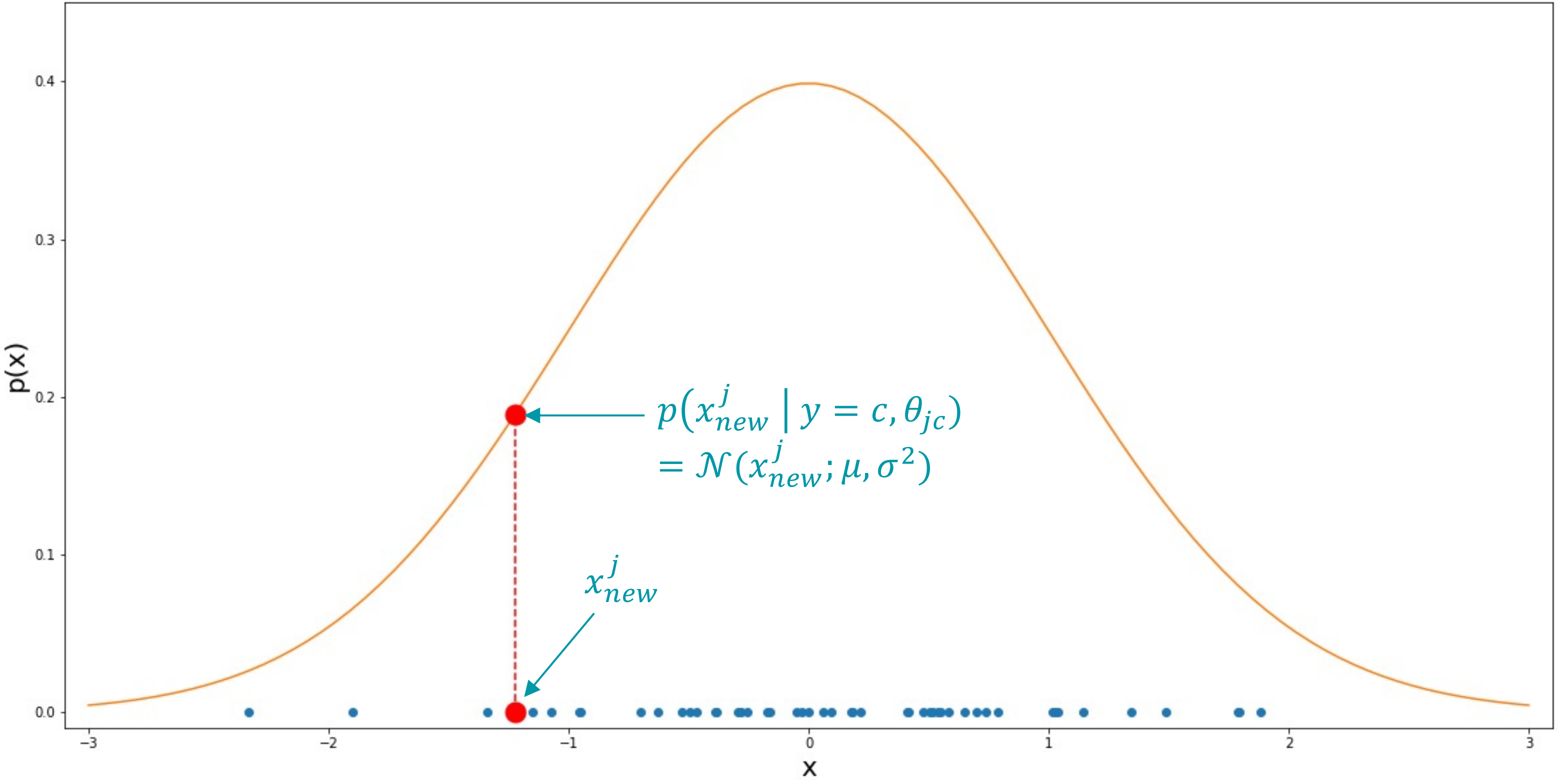
Estimation of θ_{jc}: Gaussian



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



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



Bernoulli Distribution

- The probability mass function is

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

where $x \in \{0,1\}$ and p is the probability that x = 1.

- $\theta_{jc} = p$

Multinoulli Distribution

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

- It is a special case of the *multinomial distribution*.

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

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

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

  - 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
```

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

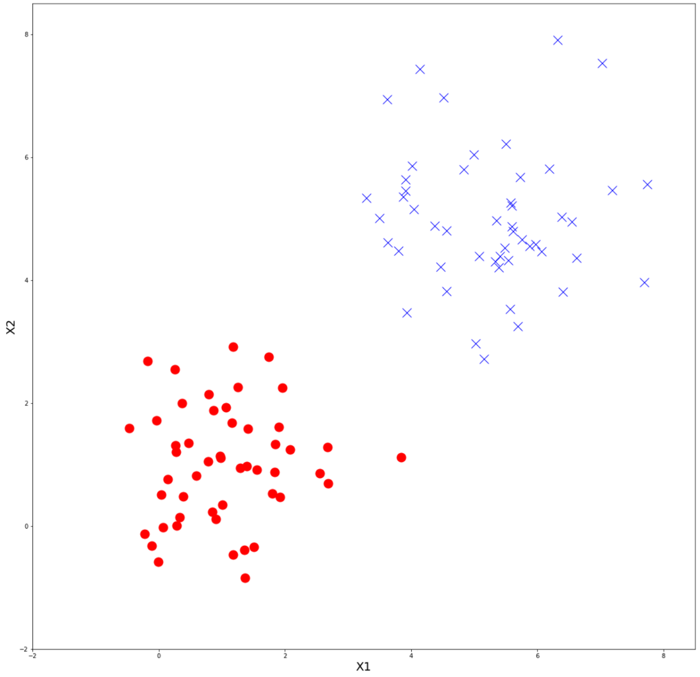
- In this pseudocode, the input of the predict function is a single data point.
- In the skeleton code, the input of the predict function is a matrix with multiple data points. The function should predict the labels for all data points.

Skeleton Code

On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes

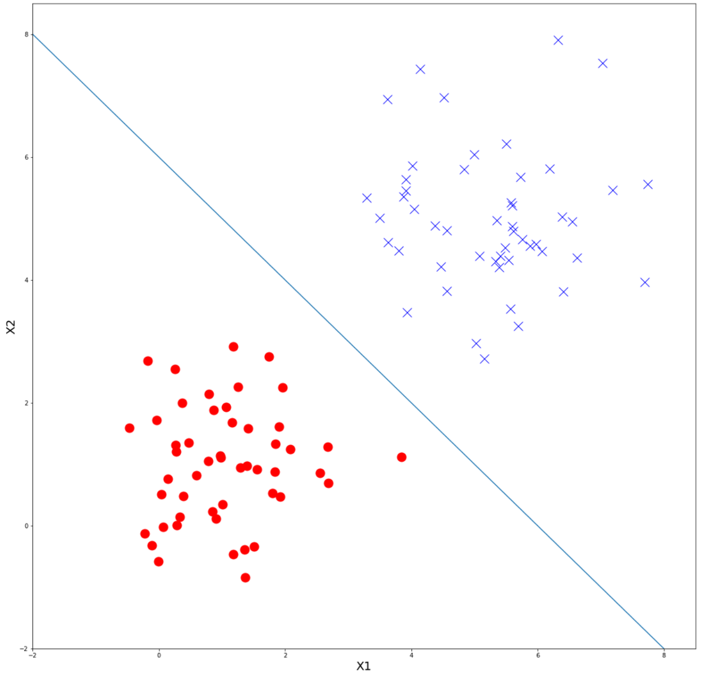
- Andrew Y. Ng and Michael I. Jordan
- NIPS’01
- Compares discriminative and generative learning as typified by logistic regression and naive bayes
- Asymptotic accuracy
    - The best accuracy the classifier can achieve given infinite training data
- Data efficiency
    - The amount of training data required to get (close) to the asymptotic accuracy

Discriminative Classifiers



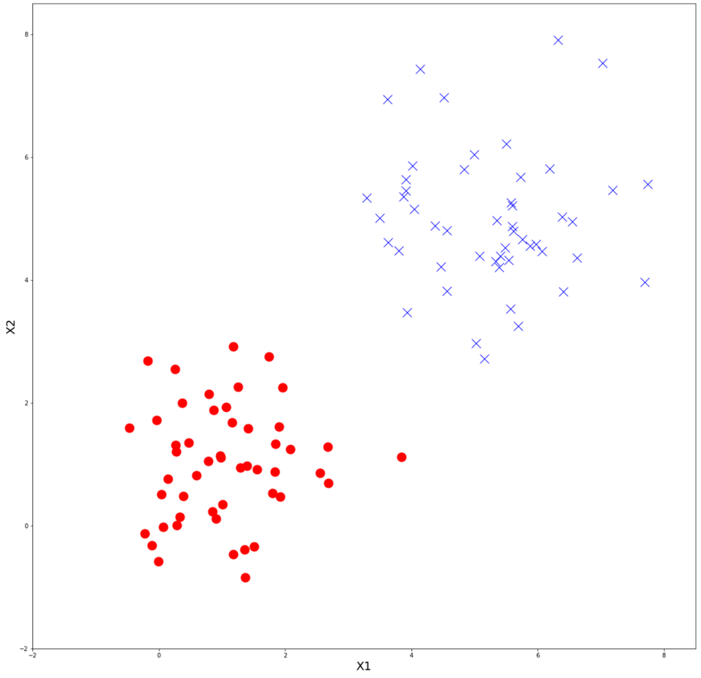
- Consider the dataset that has
  - two features X1 and X2 and
  - the labels with two classes {red, blue}
- Discriminative classifiers model **p(y | x)** using the data
  - p(y = red | x) and p(y = blue | x)

Discriminative Classifiers



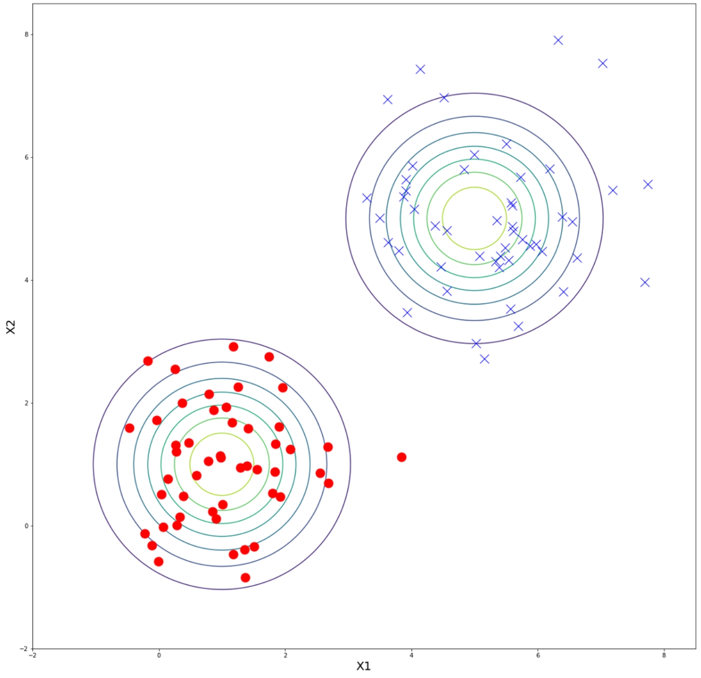
- Consider the dataset that has
  - two features X1 and X2 and
  - the labels with two classes {red, blue}
- Discriminative classifiers model **p(y | x)** using the data
  - p(y = red | x) and p(y = blue | x)
- Discriminative classifiers learn the **decision boundary** that separates the data points with two classes
  - p(y = red | x) = p(y = blue | x)

Generative Classifiers



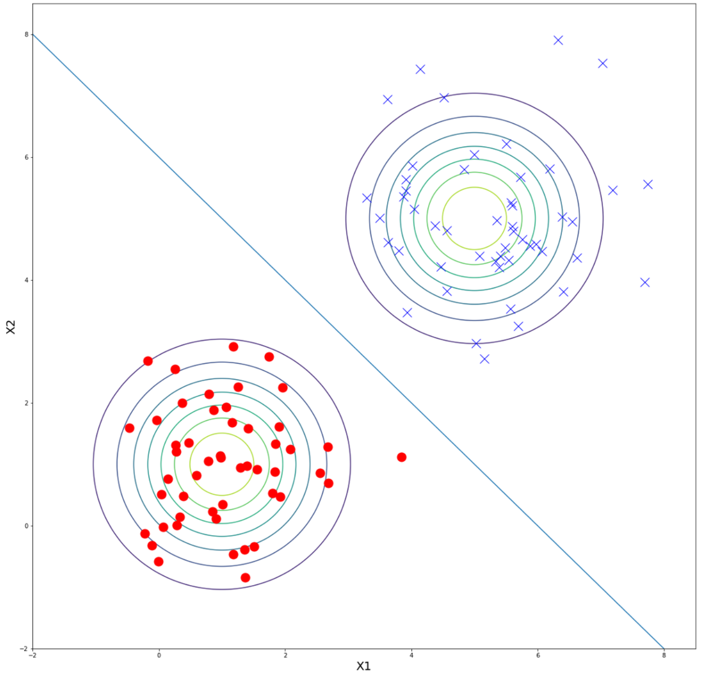
- Consider the dataset that has
  - two features X1 and X2 and
  - the labels with two classes {red, blue}
- Generative classifiers try to model **p(x,y)** (by modeling both p(x | y) and p(y)) using the data

Generative Classifiers



- Consider the dataset that has
  - two features X1 and X2 and
  - the labels with two classes {red, blue}
- Generative classifiers try to model **p(x,y)** (by modeling both p(x | y) and p(y)) using the data
  - Model the data using multivariate Gaussian distributions

Generative Classifiers



- Consider the dataset that has
  - two features X1 and X2 and
  - the labels with two classes {red, blue}
- Generative classifiers try to model **p(x,y)** (by modeling both p(x | y) and p(y)) using the data
  - Model the data using multivariate Gaussian distributions
- Apply Bayes rule to derive p(y | x):

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

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 modeling p(x,y)]” 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 modeling p(x,y)]” 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 modeling assumptions, e.g.,
  - the distribution of $p(x \mid y)$ and
  - features are conditionally independent given a class (used by naive bayes)
- Require less training data to learn "well": $O(\log D)$
  - if the assumptions are (approximately) correct
  - D is the number of parameters

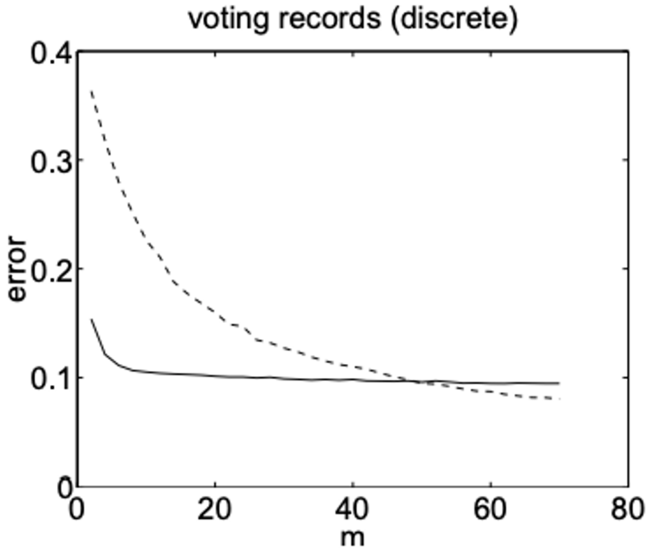
Discriminative classifiers:

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

Discriminative vs. Generative - Experiments

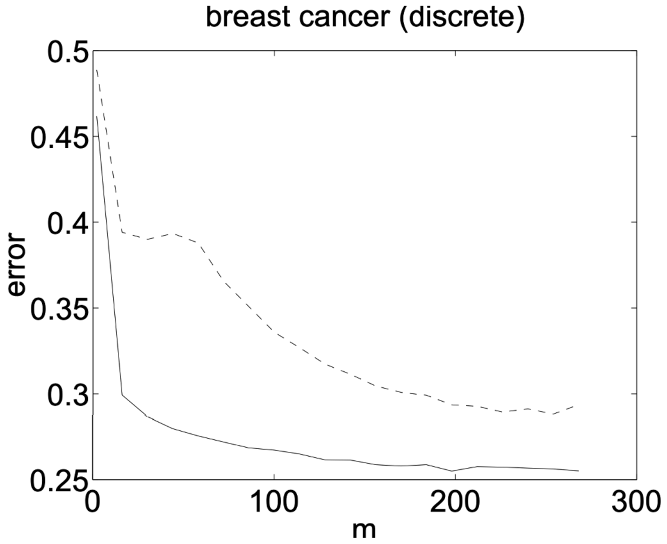
- Test logistic regression and naive bayes on 15 datasets
  - 8 with continuous inputs and
  - 7 with categorical inputs
- m: number of data points used for training
- Dashed line: logistic regression
- Solid line: naive bayes

- 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

Skeleton Code

Data Preparation

Notebook: prepare_data.ipynb

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