CSCI567 Machine Learning (Spring 2021)

Sirisha Rambhatla

University of Southern California

March 26, 2021

Outline

Outline

Logistics

- **Recognition for top teams** – Working with the department to get recognition for the top teams. *Keep up the good work!*

Outline

Density estimation

Given a training set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, **estimate a density function p that could have generated this dataset** (via $\boldsymbol{x}_n \overset{i.i.d.}{\sim} p$).

This is exactly the problem of *density estimation*, another important unsupervised learning problem.

Useful for many downstream applications such as clustering, we will look at a classification task today.

Parametric methods: generative models

Parametric estimation assumes a generative model parametrized by $\boldsymbol{\theta}$:

$$p(\boldsymbol{x}) = p(\boldsymbol{x}\,; \boldsymbol{\theta})$$

Examples:

- GMM: $p(\boldsymbol{x}\,|\boldsymbol{\theta}) = \sum_{k=1}^{K} \omega_k N(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ where $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$
- Multinomial: a discrete variable with values in $\{1, 2, \ldots, K\}$ s.t.

  $$p(x = k\,; \boldsymbol{\theta}) = \theta_k$$

  where $\boldsymbol{\theta}$ is a distribution over K elements.

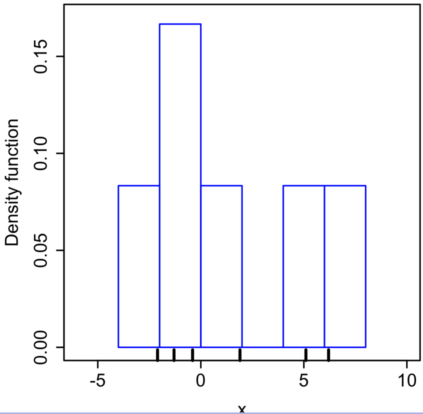
Size of $\boldsymbol{\theta}$ is independent of the training set size, so it's parametric.

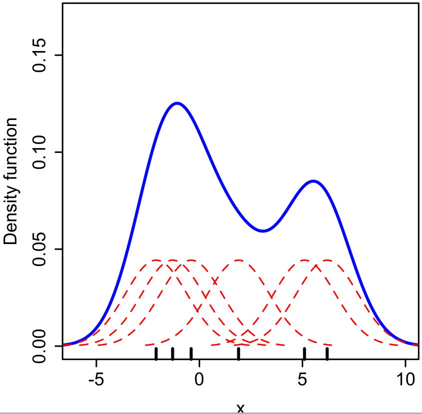
Nonparametric methods: High level idea

picture from Wikipedia

Construct something similar to a **histogram**:

- for each data point, create a "bump" (via a Kernel)
- sum up or average all the bumps





Kernel Density Estimation (KDE)

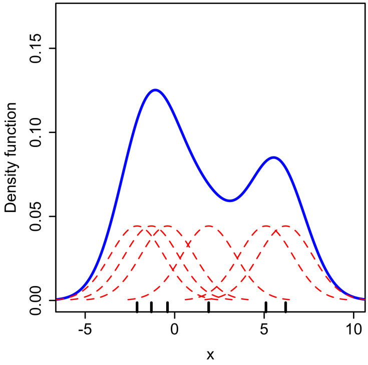
KDE with **a kernel** K: $\mathbb{R} \to \mathbb{R}$:

$$p(x) = \frac{1}{N} \sum_{n=1}^{N} K\left(x - x_n\right)$$

e.g. $K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$, the standard Gaussian density

Kernel needs to satisfy:

- symmetry: $K(u) = K(-u)$
- $\int_{-\infty}^{\infty} K(u) du = 1$, makes sure p is a density function.



Outline

1. Logistics
2. Review of last lecture: Density estimation
3. Naive Bayes
   - Setup and assumption
   - Estimation and prediction

Naive Bayes

Naive Bayes

- a simple yet surprisingly powerful **classification** algorithm

Naive Bayes

Naive Bayes

- a simple yet surprisingly powerful **classification** algorithm
- **density estimation** is one important part of the algorithm

Bayes optimal classifier

Suppose $(\boldsymbol{x}, y)$ is drawn from a joint distribution p. The **Bayes optimal classifier** is

Bayes optimal classifier

Suppose $(\boldsymbol{x}, y)$ is drawn from a joint distribution p. The **Bayes optimal classifier** is

$$f^*(\boldsymbol{x}) = \operatorname*{argmax}_{c \in [\mathsf{C}]} p(c \mid \boldsymbol{x})$$

i.e. predict the class with the largest conditional probability.

Bayes optimal classifier

Suppose $(\boldsymbol{x}, y)$ is drawn from a joint distribution p. The **Bayes optimal classifier** is

$$f^*(\boldsymbol{x}) = \operatorname*{argmax}_{c \in [\mathsf{C}]} p(c \mid \boldsymbol{x})$$

i.e. predict the class with the largest conditional probability.

p is of course unknown, but we can estimate it, which is *exactly a density estimation problem!*

Estimation

How to estimate a joint distribution? Observe we always have

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

Estimation

How to estimate a joint distribution? Observe we always have

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

We know how to estimate $p(y)$ by now.

Estimation

How to estimate a joint distribution? Observe we always have

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

We know how to estimate $p(y)$ by now.

To estimate $p(\boldsymbol{x} \mid y = c)$ for some $c \in [\mathsf{C}]$, we are doing density estimation using data $\{\boldsymbol{x}_n : y_n = c\}$.

Estimation

How to estimate a joint distribution? Observe we always have

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

We know how to estimate $p(y)$ by now.

To estimate $p(\boldsymbol{x} \mid y = c)$ for some $c \in [\mathsf{C}]$, we are doing density estimation using data $\{\boldsymbol{x}_n : y_n = c\}$.

This is *not a 1D problem* in general.

A "naive" assumption

Naive Bayes assumption:
conditioning on a label, features are independent,

A "naive" assumption

Naive Bayes assumption:
conditioning on a label, features are independent, which means

$$p(\boldsymbol{x} \mid y = c) = \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c)$$

A "naive" assumption

Naive Bayes assumption:
conditioning on a label, features are independent, which means

$$p(\boldsymbol{x} \mid y = c) = \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c)$$

Now for each d and c we have a simple 1D density estimation problem!

A "naive" assumption

Naive Bayes assumption:
conditioning on a label, features are independent, which means

$$p(\boldsymbol{x} \mid y = c) = \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c)$$

Now for each d and c we have a simple 1D density estimation problem!

Is this a reasonable assumption?

A "naive" assumption

Naive Bayes assumption:
conditioning on a label, features are independent, which means

$$p(\boldsymbol{x} \mid y = c) = \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c)$$

Now for each d and c we have a simple 1D density estimation problem!

Is this a reasonable assumption? Sometimes yes, e.g.

- use $x =$ (Height, Vocabulary) to predict $y =$ Age

A “naive” assumption

Naive Bayes assumption:

conditioning on a label, features are independent, which means

$$p(\boldsymbol{x} \mid y = c) = \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c)$$

Now for each d and c we have a simple 1D density estimation problem!

Is this a reasonable assumption? Sometimes yes, e.g.

- use $x =$ (Height, Vocabulary) to predict $y =$ Age
- Height and Vocabulary are dependent

A "naive" assumption

Naive Bayes assumption:
conditioning on a label, features are independent, which means

$$p(\boldsymbol{x} \mid y = c) = \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c)$$

Now for each d and c we have a simple 1D density estimation problem!

Is this a reasonable assumption? Sometimes yes, e.g.

- use $x = (\text{Height, Vocabulary})$ to predict $y = \text{Age}$
- Height and Vocabulary are dependent
- but condition on Age, they are independent!

A "naive" assumption

Naive Bayes assumption:
conditioning on a label, features are independent, which means

$$p(\boldsymbol{x} \mid y = c) = \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c)$$

Now for each d and c we have a simple 1D density estimation problem!

Is this a reasonable assumption? Sometimes yes, e.g.

- use $x = (\text{Height, Vocabulary})$ to predict $y = \text{Age}$
- Height and Vocabulary are dependent
- but condition on Age, they are independent!

More often this assumption is *unrealistic and "naive"*, but still Naive Bayes can work very well even if the assumption is wrong.

Example: discrete features

Height: $\leq$3', 3'-4', 4'-5', 5'-6', $\geq$6'
Vocabulary: $\leq$5K, 5K-10K, 10K-15K, 15K-20K, $\geq$20K
Age: $\leq$5, 5-10, 10-15, 15-20, 20-25, $\geq$25

Example: discrete features

Height: $\leq$3', 3'-4', 4'-5', 5'-6', $\geq$6'
Vocabulary: $\leq$5K, 5K-10K, 10K-15K, 15K-20K, $\geq$20K
Age: $\leq$5, 5-10, 10-15, 15-20, 20-25, $\geq$25

MLE estimation: e.g.

$$p(\text{Age} = \text{10-15}) = \frac{\#\text{examples with age 10-15}}{\#\text{examples}}$$

Example: discrete features

Height: ≤3', 3'-4', 4'-5', 5'-6', ≥6'
Vocabulary: ≤5K, 5K-10K, 10K-15K, 15K-20K, ≥20K
Age: ≤5, 5-10, 10-15, 15-20, 20-25, ≥25

MLE estimation: e.g.

$$p(\text{Age} = \text{10-15}) = \frac{\#\text{examples with age 10-15}}{\#\text{examples}}$$

$$p(\text{Height} = \text{5'-6'} \mid \text{Age} = \text{10-15})$$
$$= \frac{\#\text{examples with height 5'-6' and age 10-15}}{\#\text{examples with age 10-15}}$$

More formally

For a label $c \in [\mathsf{C}]$,

$$p(y = c) = \frac{|\{n : y_n = c\}|}{N}$$

More formally

For a label $c \in [\mathsf{C}]$,

$$p(y = c) = \frac{|\{n : y_n = c\}|}{N}$$

For each possible value k of a discrete feature d,

$$p(x_d = k \mid y = c) = \frac{|\{n : x_{nd} = k, y_n = c\}|}{|\{n : y_n = c\}|}$$

Continuous features

If the feature is continuous, we can do

- parametric estimation,

- or nonparametric estimation,

Continuous features

If the feature is continuous, we can do

- parametric estimation, e.g. via a Gaussian

$$p(x_d = x \mid y = c) = \frac{1}{\sqrt{2\pi}\sigma_{cd}} \exp\left(-\frac{(x-\mu_{cd})^2}{2\sigma_{cd}^2}\right)$$

- or nonparametric estimation,

Continuous features

If the feature is continuous, we can do

- parametric estimation, e.g. via a Gaussian

$$p(x_d = x \mid y = c) = \frac{1}{\sqrt{2\pi}\sigma_{cd}} \exp\left(-\frac{(x-\mu_{cd})^2}{2\sigma_{cd}^2}\right)$$

  where μ_{cd} and σ_{cd}^2 are the empirical mean and variance of feature d among all examples with label c. Notice connections to Linear Discriminant Analysis (LDA)!

- or nonparametric estimation,

Continuous features

If the feature is continuous, we can do

- parametric estimation, e.g. via a Gaussian

$$p(x_d = x \mid y = c) = \frac{1}{\sqrt{2\pi}\sigma_{cd}} \exp\left(-\frac{(x-\mu_{cd})^2}{2\sigma_{cd}^2}\right)$$

  where μ_{cd} and σ_{cd}^2 are the empirical mean and variance of feature d among all examples with label c. Notice connections to Linear Discriminant Analysis (LDA)!

- or nonparametric estimation, e.g. via a Kernel K and bandwidth h:

$$p(x_d = x \mid y = c) = \frac{1}{|\{n : y_n = c\}|} \sum_{n:y_n=c} K_h(x - x_{nd})$$

How to predict?

After learning the model

$$p(\boldsymbol{x}, y) = p(y) \prod_{d=1}^{\mathsf{D}} p(x_d \mid y)$$

How to predict?

After learning the model

$$p(\boldsymbol{x}, y) = p(y) \prod_{d=1}^{\mathsf{D}} p(x_d \mid y)$$

the **prediction** for a new example $\boldsymbol{x}$ is

$$\underset{c \in [\mathsf{C}]}{\operatorname{argmax}} \; p(y = c \mid \boldsymbol{x})$$

How to predict?

After learning the model

$$p(\boldsymbol{x}, y) = p(y) \prod_{d=1}^{\mathsf{D}} p(x_d \mid y)$$

the **prediction** for a new example $\boldsymbol{x}$ is

$$\operatorname*{argmax}_{c \in [\mathsf{C}]} \; p(y = c \mid \boldsymbol{x}) = \operatorname*{argmax}_{c \in [\mathsf{C}]} \; p(\boldsymbol{x}, y = c)$$

How to predict?

After learning the model

$$p(\boldsymbol{x}, y) = p(y) \prod_{d=1}^{\mathsf{D}} p(x_d \mid y)$$

the **prediction** for a new example $\boldsymbol{x}$ is

$$\begin{aligned} \operatorname*{argmax}_{c \in [\mathsf{C}]} \; p(y = c \mid \boldsymbol{x}) &= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; p(\boldsymbol{x}, y = c) \\ &= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(p(y = c) \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c) \right) \end{aligned}$$

How to predict?

After learning the model

$$p(\boldsymbol{x}, y) = p(y) \prod_{d=1}^{\mathsf{D}} p(x_d \mid y)$$

the **prediction** for a new example $\boldsymbol{x}$ is

$$\begin{aligned}
\operatorname*{argmax}_{c \in [\mathsf{C}]} \; p(y = c \mid \boldsymbol{x}) &= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; p(\boldsymbol{x}, y = c) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(p(y = c) \prod_{d=1}^{\mathsf{D}} p(x_d \mid y = c) \right) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(\ln p(y = c) + \sum_{d=1}^{\mathsf{D}} \ln p(x_d \mid y = c) \right)
\end{aligned}$$

Examples

For **discrete features**, plugging in previous MLE estimations gives

$$
\begin{aligned}
&\operatorname*{argmax}_{c \in [\mathsf{C}]} \; p(y = c \mid \boldsymbol{x}) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(\ln p(y = c) + \sum_{d=1}^{\mathsf{D}} \ln p(x_d \mid y = c) \right) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(\ln |\{n : y_n = c\}| + \sum_{d=1}^{\mathsf{D}} \ln \frac{|\{n : x_{nd} = x_d, y_n = c\}|}{|\{n : y_n = c\}|} \right)
\end{aligned}
$$

Examples

For **continuous features** with a Gaussian model,

$$
\begin{aligned}
&\operatorname*{argmax}_{c\in[\mathsf{C}]}\ p(y=c \mid \boldsymbol{x}) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \left(\ln p(y=c) + \sum_{d=1}^{\mathsf{D}} \ln p(x_d \mid y=c) \right) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \left(\ln |\{n : y_n = c\}| + \sum_{d=1}^{\mathsf{D}} \ln \left(\frac{1}{\sqrt{2\pi}\sigma_{cd}} \exp\left(-\frac{(x_d-\mu_{cd})^2}{2\sigma_{cd}^2} \right) \right) \right)
\end{aligned}
$$

Examples

For **continuous features** with a Gaussian model,

$$
\begin{aligned}
&\operatorname*{argmax}_{c\in[\mathsf{C}]} \; p(y=c \mid \boldsymbol{x}) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \; \left(\ln p(y=c) + \sum_{d=1}^{\mathsf{D}} \ln p(x_d \mid y=c) \right) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \; \left(\ln |\{n : y_n = c\}| + \sum_{d=1}^{\mathsf{D}} \ln \left(\frac{1}{\sqrt{2\pi}\sigma_{cd}} \exp\left(-\frac{(x_d-\mu_{cd})^2}{2\sigma_{cd}^2} \right) \right) \right) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \; \left(\ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \left(\ln \sigma_{cd} + \frac{(x_d-\mu_{cd})^2}{2\sigma_{cd}^2} \right) \right)
\end{aligned}
$$

Examples

For **continuous features** with a Gaussian model,

$$
\begin{aligned}
&\operatorname*{argmax}_{c\in[\mathsf{C}]} \; p(y=c \mid \boldsymbol{x}) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \left(\ln p(y=c) + \sum_{d=1}^{\mathsf{D}} \ln p(x_d \mid y=c) \right) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \left(\ln |\{n : y_n = c\}| + \sum_{d=1}^{\mathsf{D}} \ln \left(\frac{1}{\sqrt{2\pi}\sigma_{cd}} \exp\left(-\frac{(x_d-\mu_{cd})^2}{2\sigma_{cd}^2} \right) \right) \right) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]} \left(\ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \left(\ln \sigma_{cd} + \frac{(x_d-\mu_{cd})^2}{2\sigma_{cd}^2} \right) \right)
\end{aligned}
$$

which is *quadratic* in the feature $\boldsymbol{x}$.

What is naive Bayes learning?

Observe again the case for continuous features with a Gaussian model, if we **fix the variance for each feature to be σ^2** (i.e. not a parameter of the model any more), then the prediction becomes

$$\begin{aligned}
&\operatorname*{argmax}_{c\in[\mathsf{C}]}\; p(y = c \mid \boldsymbol{x}) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]}\; \left(\ln|\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}}\left(\ln\sigma + \frac{(x_d - \mu_{cd})^2}{2\sigma^2}\right)\right)
\end{aligned}$$

What is naive Bayes learning?

Observe again the case for continuous features with a Gaussian model, if we **fix the variance for each feature to be σ^2** (i.e. not a parameter of the model any more), then the prediction becomes

$$
\begin{aligned}
&\operatorname*{argmax}_{c\in[\mathsf{C}]}\ p(y=c \mid \boldsymbol{x}) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]}\ \left(\ln|\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}}\left(\ln\sigma + \frac{(x_d-\mu_{cd})^2}{2\sigma^2}\right)\right) \\
&= \operatorname*{argmax}_{c\in[\mathsf{C}]}\ \left(\ln|\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}}\frac{\mu_{cd}^2}{2\sigma^2} + \sum_{d=1}^{\mathsf{D}}\frac{\mu_{cd}}{\sigma^2}x_d\right)
\end{aligned}
$$

What is naive Bayes learning?

Observe again the case for continuous features with a Gaussian model, if we **fix the variance for each feature to be σ^2** (i.e. not a parameter of the model any more), then the prediction becomes

$$
\begin{aligned}
&\operatorname*{argmax}_{c \in [\mathsf{C}]} \; p(y = c \mid \boldsymbol{x}) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(\ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \left(\ln \sigma + \frac{(x_d - \mu_{cd})^2}{2\sigma^2} \right) \right) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(\ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \frac{\mu_{cd}^2}{2\sigma^2} + \sum_{d=1}^{\mathsf{D}} \frac{\mu_{cd}}{\sigma^2} x_d \right) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(w_{c0} + \sum_{d=1}^{\mathsf{D}} w_{cd} x_d \right)
\end{aligned}
$$

where we denote $w_{c0} = \ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \frac{\mu_{cd}^2}{2\sigma^2}$ and $w_{cd} = \frac{\mu_{cd}}{\sigma^2}$.

What is naive Bayes learning?

Observe again the case for continuous features with a Gaussian model, if we **fix the variance for each feature to be σ^2** (i.e. not a parameter of the model any more), then the prediction becomes

$$
\begin{aligned}
&\operatorname*{argmax}_{c \in [\mathsf{C}]} \; p(y = c \mid \boldsymbol{x}) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(\ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \left(\ln \sigma + \frac{(x_d - \mu_{cd})^2}{2\sigma^2} \right) \right) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(\ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \frac{\mu_{cd}^2}{2\sigma^2} + \sum_{d=1}^{\mathsf{D}} \frac{\mu_{cd}}{\sigma^2} x_d \right) \\
&= \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \left(w_{c0} + \sum_{d=1}^{\mathsf{D}} w_{cd} x_d \right) = \operatorname*{argmax}_{c \in [\mathsf{C}]} \; \boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x} \qquad (\textit{linear classifier!})
\end{aligned}
$$

where we denote $w_{c0} = \ln |\{n : y_n = c\}| - \sum_{d=1}^{\mathsf{D}} \frac{\mu_{cd}^2}{2\sigma^2}$ and $w_{cd} = \frac{\mu_{cd}}{\sigma^2}$.

Connection to logistic regression

Moreover by similar calculation one can verify

$$p(y = c \mid \boldsymbol{x}) \propto e^{\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x}}$$

Connection to logistic regression

Moreover by similar calculation one can verify

$$p(y = c \mid \boldsymbol{x}) \propto e^{\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x}}$$

This is exactly the **softmax** function, *the same model we used for a probabilistic interpretation of logistic regression!*

Connection to logistic regression

Moreover by similar calculation one can verify

$$p(y = c \mid \boldsymbol{x}) \propto e^{\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x}}$$

This is exactly the **softmax** function, *the same model we used for a probabilistic interpretation of logistic regression!*

So what is different then?

Connection to logistic regression

Moreover by similar calculation one can verify

$$p(y = c \mid \boldsymbol{x}) \propto e^{\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x}}$$

This is exactly the **softmax** function, *the same model we used for a probabilistic interpretation of logistic regression!*

So what is different then? They **learn the parameters in different ways**:

Connection to logistic regression

Moreover by similar calculation one can verify

$$p(y = c \mid \boldsymbol{x}) \propto e^{\boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{x}}$$

This is exactly the **softmax** function, *the same model we used for a probabilistic interpretation of logistic regression!*

So what is different then? They **learn the parameters in different ways**:

- both via MLE, one on $p(y = c \mid x)$, the other on $p(x, y)$

Connection to logistic regression

Moreover by similar calculation one can verify

$$p(y = c \mid \boldsymbol{x}) \propto e^{\boldsymbol{w}_c^{\mathrm{T}}\boldsymbol{x}}$$

This is exactly the **softmax** function, *the same model we used for a probabilistic interpretation of logistic regression!*

So what is different then? They **learn the parameters in different ways**:

- both via MLE, one on $p(y = c \mid x)$, the other on $p(x, y)$
- solutions are different: logistic regression has no closed-form, naive Bayes admits a simple closed-form

Generative model v.s discriminative model

	Discriminative model	Generative model
Example	logistic regression	naive Bayes

Generative model v.s discriminative model

	Discriminative model	Generative model
Example	logistic regression	naive Bayes
Model	conditional $p(y \mid x)$	joint $p(x, y)$ (might have same $p(y \mid x)$)

Generative model v.s discriminative model

	Discriminative model	Generative model
Example	logistic regression	naive Bayes
Model	conditional $p(y \mid x)$	joint $p(x, y)$ (might have same $p(y \mid x)$)
Learning	MLE	MLE

Generative model v.s discriminative model

	Discriminative model	Generative model
Example	logistic regression	naive Bayes
Model	conditional $p(y \mid x)$	joint $p(x, y)$ (might have same $p(y \mid x)$)
Learning	MLE	MLE
Accuracy	usually better for large N	usually better for small N

Generative model v.s discriminative model

	Discriminative model	Generative model
Example	logistic regression	naive Bayes
Model	conditional $p(y \mid x)$	joint $p(x, y)$ (might have same $p(y \mid x)$)
Learning	MLE	MLE
Accuracy	usually better for large N	usually better for small N
Remark		more flexible, can generate data after learning