Chapter 4: Bayesian Learning

Reza Rezazadegan

Sharif University of Technology

The idea behind Bayesian learning

- We have seen that an ML model in a given family (e.g. Logistic Regression or Neural Networks) is determined by its parameters

The idea behind Bayesian learning

- We have seen that an ML model in a given family (e.g. Logistic Regression or Neural Networks) is determined by its parameters and that the values of the parameters are determined in the training process.

The idea behind Bayesian learning

- We have seen that an ML model in a given family (e.g. Logistic Regression or Neural Networks) is determined by its parameters and that the values of the parameters are determined in the training process.
- The basic idea behind Bayesian Learning is that model parameters instead of having deterministic values, are given by a probability distribution.

The idea behind Bayesian learning

- We have seen that an ML model in a given family (e.g. Logistic Regression or Neural Networks) is determined by its parameters and that the values of the parameters are determined in the training process.
- The basic idea behind Bayesian Learning is that model parameters instead of having deterministic values, are given by a probability distribution.
- Thus Bayesian Learning is inherently more complex than other methods.

The idea behind Bayesian learning

- We have seen that an ML model in a given family (e.g. Logistic Regression or Neural Networks) is determined by its parameters and that the values of the parameters are determined in the training process.
- The basic idea behind Bayesian Learning is that model parameters instead of having deterministic values, are given by a probability distribution.
- Thus Bayesian Learning is inherently more complex than other methods.
- **The Frequentist view of statistics:** probability is the limit of the frequency of occurrence when the number of samples goes towards infinity.

The idea behind Bayesian learning

- We have seen that an ML model in a given family (e.g. Logistic Regression or Neural Networks) is determined by its parameters and that the values of the parameters are determined in the training process.
- The basic idea behind Bayesian Learning is that model parameters instead of having deterministic values, are given by a probability distribution.
- Thus Bayesian Learning is inherently more complex than other methods.
- **The Frequentist view of statistics:** probability is the limit of the frequency of occurrence when the number of samples goes towards infinity.
- **The Bayesian view of statistics:** probability is a measure of belief in the occurrence of events.

The idea behind Bayesian learning

- The **Bayes Rule** is the cornerstone of Bayesian learning:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \tag{1}$$

The idea behind Bayesian learning

- The **Bayes Rule** is the cornerstone of Bayesian learning:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \tag{1}$$

- Here H is interpreted as a hypothesis (e.g. our model parameters) and D is the frequency distribution of the data.
- $P(H|D)$: **Posterior**

The idea behind Bayesian learning

- The **Bayes Rule** is the cornerstone of Bayesian learning:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \tag{1}$$

- Here H is interpreted as a hypothesis (e.g. our model parameters) and D is the frequency distribution of the data.
- $P(H|D)$: **Posterior**
- $P(H)$: **Prior**

The idea behind Bayesian learning

- The **Bayes Rule** is the cornerstone of Bayesian learning:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \tag{1}$$

- Here H is interpreted as a hypothesis (e.g. our model parameters) and D is the frequency distribution of the data.
- $P(H|D)$: **Posterior**
- $P(H)$: **Prior**
- $P(D|H)$: **Likelihood**

The idea behind Bayesian learning

- The **Bayes Rule** is the cornerstone of Bayesian learning:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \tag{1}$$

- Here H is interpreted as a hypothesis (e.g. our model parameters) and D is the frequency distribution of the data.
- $P(H|D)$: **Posterior**
- $P(H)$: **Prior**
- $P(D|H)$: **Likelihood**
- $P(D)$: **Evidence**

The idea behind Bayesian learning

- The **Bayes Rule** is the cornerstone of Bayesian learning:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \tag{1}$$

- Here H is interpreted as a hypothesis (e.g. our model parameters) and D is the frequency distribution of the data.
- $P(H|D)$: **Posterior**
- $P(H)$: **Prior**
- $P(D|H)$: **Likelihood**
- $P(D)$: **Evidence**
- **Prior believes influence posterior believes!**

The Naive Bayes Classifier

- Let $x_1, x_2, \ldots, x_d$ be our features, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and $C_1, C_2, \ldots, C_m$ be the classes.

The Naive Bayes Classifier

- Let $x_1, x_2, \ldots, x_d$ be our features, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and $C_1, C_2, \ldots, C_m$ be the classes. We want to know the *posterior* probability of $\mathbf{x}$ belonging to C_k i.e. $P(C_k|\mathbf{x})$.

The Naive Bayes Classifier

- Let $x_1, x_2, \ldots, x_d$ be our features, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and $C_1, C_2, \ldots, C_m$ be the classes. We want to know the *posterior* probability of $\mathbf{x}$ belonging to C_k i.e. $P(C_k|\mathbf{x})$. The Bayes rule tells us:

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})}. \tag{2}$$

The Naive Bayes Classifier

- Let $x_1, x_2, \ldots, x_d$ be our features, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and $C_1, C_2, \ldots, C_m$ be the classes. We want to know the *posterior* probability of $\mathbf{x}$ belonging to C_k i.e. $P(C_k|\mathbf{x})$. The Bayes rule tells us:

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})}. \tag{2}$$

- The *Prior* $P(C_k)$ is the fraction of datapoints belonging to the class C_k.

The Naive Bayes Classifier

- Let $x_1, x_2, \ldots, x_d$ be our features, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and $C_1, C_2, \ldots, C_m$ be the classes. We want to know the *posterior* probability of $\mathbf{x}$ belonging to C_k i.e. $P(C_k|\mathbf{x})$. The Bayes rule tells us:

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})}. \tag{2}$$

- The *Prior* $P(C_k)$ is the fraction of datapoints belonging to the class C_k.
- The *likelihood* $P(\mathbf{x}|C_k)$ can be computed using the "naive" *Conditional Independence* hypothesis:

The Naive Bayes Classifier

- Let $x_1, x_2, \ldots, x_d$ be our features, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and $C_1, C_2, \ldots, C_m$ be the classes. We want to know the *posterior* probability of $\mathbf{x}$ belonging to C_k i.e. $P(C_k|\mathbf{x})$. The Bayes rule tells us:

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})}. \tag{2}$$

- The *Prior* $P(C_k)$ is the fraction of datapoints belonging to the class C_k.
- The *likelihood* $P(\mathbf{x}|C_k)$ can be computed using the "naive" *Conditional Independence* hypothesis:

$$P(\mathbf{x}|C_k) = P(x_1|C_k)P(x_2|C_k)\cdots P(x_d|C_k). \tag{3}$$

We are assuming that features in each class are independent of each other!

The Naive Bayes Classifier

- Let $x_1, x_2, \ldots, x_d$ be our features, $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and $C_1, C_2, \ldots, C_m$ be the classes. We want to know the *posterior* probability of $\mathbf{x}$ belonging to C_k i.e. $P(C_k|\mathbf{x})$. The Bayes rule tells us:
$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})}. \tag{2}$$
- The *Prior* $P(C_k)$ is the fraction of datapoints belonging to the class C_k.
- The *likelihood* $P(\mathbf{x}|C_k)$ can be computed using the "naive" *Conditional Independence* hypothesis:
$$P(\mathbf{x}|C_k) = P(x_1|C_k)P(x_2|C_k)\cdots P(x_d|C_k). \tag{3}$$
We are assuming that features in each class are independent of each other!
- Since, $P(\mathbf{x})$ is independent of C_k and we are looking for a class k that maximized $P(C_k|\mathbf{x})$, we can drop $P(\mathbf{x})$. (Alternatively, consider $P(C_k|\mathbf{x})/P(C_l|\mathbf{x})$).

The Naive Bayes Classifier cont.

- The probabilities $P(x_i|C_k)$ can be estimated by assuming that the feature values follow a probability distribution (such as normal or Bernoulli distribution) and using training data to estimate the parameters of the distribution.

The Naive Bayes Classifier cont.

- The probabilities $P(x_i|C_k)$ can be estimated by assuming that the feature values follow a probability distribution (such as normal or Bernoulli distribution) and using training data to estimate the parameters of the distribution.
- For example for continuous features (quantities), such as height, IQ, etc, often follow a **Gaussian (normal) distribution**:

The Naive Bayes Classifier cont.

- The probabilities $P(x_i|C_k)$ can be estimated by assuming that the feature values follow a probability distribution (such as normal or Bernoulli distribution) and using training data to estimate the parameters of the distribution.
- For example for continuous features (quantities), such as height, IQ, etc, often follow a **Gaussian (normal) distribution**:

$$P(x_i = v) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(v-\mu)^2}{2\sigma^2}} \tag{4}$$

where μ, σ are the mean and standard deviation of the feature x_i in the class C_k

The Naive Bayes Classifier cont.

- The probabilities $P(x_i|C_k)$ can be estimated by assuming that the feature values follow a probability distribution (such as normal or Bernoulli distribution) and using training data to estimate the parameters of the distribution.
- For example for continuous features (quantities), such as height, IQ, etc, often follow a **Gaussian (normal) distribution**:

$$P(x_i = v) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(v-\mu)^2}{2\sigma^2}} \tag{4}$$

where μ, σ are the mean and standard deviation of the feature x_i in the class C_k and can be estimated from the data.

The Naive Bayes Classifier cont.

- The probabilities $P(x_i|C_k)$ can be estimated by assuming that the feature values follow a probability distribution (such as normal or Bernoulli distribution) and using training data to estimate the parameters of the distribution.
- For example for continuous features (quantities), such as height, IQ, etc, often follow a **Gaussian (normal) distribution**:

$$P(x_i = v) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(v-\mu)^2}{2\sigma^2}} \tag{4}$$

  where μ, σ are the mean and standard deviation of the feature x_i in the class C_k and can be estimated from the data.
- For each $\mathbf{x}$, the class C_k that maximizes $p(C_i|\mathbf{x})$ among different C_i, is chosen as the predicted class of $\mathbf{x}$.

An example of using Naive Bayes Classifier

Consider the following dataset:

Person	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

Use the Naive Bayes Classifier with Gaussian distribution to predict the gender of a person whose height, weight and foot size are, respectively, 6, 130, 8.

The Naive Bayes Classifier cont.

- The **Bernoulli distribution** can be used for binary-valued features $x_i \in \{0, 1\}$ (e.g. whether a word occurs in a short text or not):

The Naive Bayes Classifier cont.

- The **Bernoulli distribution** can be used for binary-valued features $x_i \in \{0, 1\}$ (e.g. whether a word occurs in a short text or not):

$$P(\mathbf{x}|C_k) = \prod_{i=1}^{d} p_i^{x_i}(1 - p_i)^{1-x_i} \tag{5}$$

where p_i is the probability that the class k involves x_i.

The Naive Bayes Classifier cont.

- The **Bernoulli distribution** can be used for binary-valued features $x_i \in \{0, 1\}$ (e.g. whether a word occurs in a short text or not):

$$P(\mathbf{x}|C_k) = \prod_{i=1}^{d} p_i^{x_i}(1 - p_i)^{1-x_i} \tag{5}$$

  where p_i is the probability that the class k involves x_i.
- For example imagine the the two classes are *spam* and *ham*, and our training data consists of short emails labeled by these two classes.

The Naive Bayes Classifier cont.

- The **Bernoulli distribution** can be used for binary-valued features $x_i \in \{0, 1\}$ (e.g. whether a word occurs in a short text or not):

$$P(\mathbf{x}|C_k) = \prod_{i=1}^{d} p_i^{x_i}(1 - p_i)^{1-x_i} \tag{5}$$

  where p_i is the probability that the class k involves x_i.
- For example imagine the the two classes are *spam* and *ham*, and our training data consists of short emails labeled by these two classes.
- If $w_1, w_2, \ldots, w_d$ represent the combined vocabulary of the training texts, then the feature $x_i(T)$, of a document T, is 0 or 1 based on whether the word w_i occurs in T or not.

The Naive Bayes Classifier cont.

- The **Bernoulli distribution** can be used for binary-valued features $x_i \in \{0, 1\}$ (e.g. whether a word occurs in a short text or not):

$$P(\mathbf{x}|C_k) = \prod_{i=1}^{d} p_i^{x_i}(1 - p_i)^{1-x_i} \tag{5}$$

  where p_i is the probability that the class k involves x_i.
- For example imagine the the two classes are *spam* and *ham*, and our training data consists of short emails labeled by these two classes.
- If $w_1, w_2, \ldots, w_d$ represent the combined vocabulary of the training texts, then the feature $x_i(T)$, of a document T, is 0 or 1 based on whether the word w_i occurs in T or not.
- Then, p_i for the class *spam* is the probability of spam emails containing the word w_i.

The Naive Bayes Classifier cont.

- The **Bernoulli distribution** can be used for binary-valued features $x_i \in \{0, 1\}$ (e.g. whether a word occurs in a short text or not):

$$P(\mathbf{x}|C_k) = \prod_{i=1}^{d} p_i^{x_i}(1 - p_i)^{1-x_i} \tag{5}$$

  where p_i is the probability that the class k involves x_i.
- For example imagine the the two classes are *spam* and *ham*, and our training data consists of short emails labeled by these two classes.
- If $w_1, w_2, \ldots, w_d$ represent the combined vocabulary of the training texts, then the feature $x_i(T)$, of a document T, is 0 or 1 based on whether the word w_i occurs in T or not.
- Then, p_i for the class *spam* is the probability of spam emails containing the word w_i. It is simply the fraction of training spam emails which contain this word!

Naive Bayes Classifier with Multinomial Distribution

- The Multinomial Distribution is used when features x_i are counts, e.g. the count of words in documents.

Naive Bayes Classifier with Multinomial Distribution

- The Multinomial Distribution is used when features x_i are counts, e.g. the count of words in documents.
- In Natural Language Processing, a document is often regarded as a *bag of words*, regardless of the order of words in it.

Naive Bayes Classifier with Multinomial Distribution

- The Multinomial Distribution is used when features x_i are counts, e.g. the count of words in documents.
- In Natural Language Processing, a document is often regarded as a *bag of words*, regardless of the order of words in it.
- If p_i is the probability of feature i occurring in class C_k then the Multinomial Distribution gives us:

Naive Bayes Classifier with Multinomial Distribution

- The Multinomial Distribution is used when features x_i are counts, e.g. the count of words in documents.
- In Natural Language Processing, a document is often regarded as a *bag of words*, regardless of the order of words in it.
- If p_i is the probability of feature i occurring in class C_k then the Multinomial Distribution gives us:

$$P(\mathbf{x}|C_k) = \frac{(x_1 + x_2 + \cdots + x_d)!}{x_1!x_2!\dots x_d!} p_1^{x_1} p_2^{x_2} \dots p_d^{x_d} \tag{6}$$

Naive Bayes Classifier with Multinomial Distribution

- The Multinomial Distribution is used when features x_i are counts, e.g. the count of words in documents.
- In Natural Language Processing, a document is often regarded as a *bag of words*, regardless of the order of words in it.
- If p_i is the probability of feature i occurring in class C_k then the Multinomial Distribution gives us:

$$P(\mathbf{x}|C_k) = \frac{(x_1 + x_2 + \cdots + x_d)!}{x_1!x_2!\dots x_d!} p_1^{x_1} p_2^{x_2} \dots p_d^{x_d} \tag{6}$$

- In text classification, the feature $x_i(T)$ of a document T, is the count of the word w_i in T.

Naive Bayes Classifier with Multinomial Distribution

- The Multinomial Distribution is used when features x_i are counts, e.g. the count of words in documents.
- In Natural Language Processing, a document is often regarded as a *bag of words*, regardless of the order of words in it.
- If p_i is the probability of feature i occurring in class C_k then the Multinomial Distribution gives us:

$$P(\mathbf{x}|C_k) = \frac{(x_1 + x_2 + \cdots + x_d)!}{x_1!x_2!\ldots x_d!} p_1^{x_1} p_2^{x_2} \ldots p_d^{x_d} \tag{6}$$

- In text classification, the feature $x_i(T)$ of a document T, is the count of the word w_i in T.
- The probabilities p_i are approximated by the number of times, in our training data, the feature i (e.g. the word w_i) appears in class C_k, divided by the number of all features (e.g. words) in the class C_k.

Naive Bayes Classifier with Multinomial Distribution

- The Multinomial Distribution is used when features x_i are counts, e.g. the count of words in documents.
- In Natural Language Processing, a document is often regarded as a *bag of words*, regardless of the order of words in it.
- If p_i is the probability of feature i occurring in class C_k then the Multinomial Distribution gives us:

$$P(\mathbf{x}|C_k) = \frac{(x_1 + x_2 + \cdots + x_d)!}{x_1!x_2!\ldots x_d!} p_1^{x_1} p_2^{x_2} \ldots p_d^{x_d} \tag{6}$$

- In text classification, the feature $x_i(T)$ of a document T, is the count of the word w_i in T.
- The probabilities p_i are approximated by the number of times, in our training data, the feature i (e.g. the word w_i) appears in class C_k, divided by the number of all features (e.g. words) in the class C_k.
- Multinomial Naive Bayes does not assume conditional independence!

Pros and Cons of Naive Bayes Classifier

- Even though Conditional Independence is a strong assumption, Naive Bayes is a powerful classifier.

Pros and Cons of Naive Bayes Classifier

- Even though Conditional Independence is a strong assumption, Naive Bayes is a powerful classifier.
- Naive Bayes is a probabilistic, multi-class classifier. It needs a small amount of training data.

Pros and Cons of Naive Bayes Classifier

- Even though Conditional Independence is a strong assumption, Naive Bayes is a powerful classifier.
- Naive Bayes is a probabilistic, multi-class classifier. It needs a small amount of training data.
- The conditional independence assumption helps with high dimensional data (i.e. a lot of features).

Pros and Cons of Naive Bayes Classifier

- Even though Conditional Independence is a strong assumption, Naive Bayes is a powerful classifier.
- Naive Bayes is a probabilistic, multi-class classifier. It needs a small amount of training data.
- The conditional independence assumption helps with high dimensional data (i.e. a lot of features).
- Naive Bayes works particularly well in document classification and spam filtering.

Pros and Cons of Naive Bayes Classifier

- Even though Conditional Independence is a strong assumption, Naive Bayes is a powerful classifier.
- Naive Bayes is a probabilistic, multi-class classifier. It needs a small amount of training data.
- The conditional independence assumption helps with high dimensional data (i.e. a lot of features).
- Naive Bayes works particularly well in document classification and spam filtering.
- Naive Bayes can be used for online learning, as it is easy to update distribution parameters according to new data.

Pros and Cons of Naive Bayes Classifier

- Even though Conditional Independence is a strong assumption, Naive Bayes is a powerful classifier.
- Naive Bayes is a probabilistic, multi-class classifier. It needs a small amount of training data.
- The conditional independence assumption helps with high dimensional data (i.e. a lot of features).
- Naive Bayes works particularly well in document classification and spam filtering.
- Naive Bayes can be used for online learning, as it is easy to update distribution parameters according to new data.
- Even though NB is a good classifier, the probabilities it produces are not precise.

Pros and Cons of Naive Bayes Classifier

- Even though Conditional Independence is a strong assumption, Naive Bayes is a powerful classifier.
- Naive Bayes is a probabilistic, multi-class classifier. It needs a small amount of training data.
- The conditional independence assumption helps with high dimensional data (i.e. a lot of features).
- Naive Bayes works particularly well in document classification and spam filtering.
- Naive Bayes can be used for online learning, as it is easy to update distribution parameters according to new data.
- Even though NB is a good classifier, the probabilities it produces are not precise.
- In Scikit-Learn, Naive Bayes is provided through classes `MultinomialNB`, `BernoulliNB` and `GaussianNB`

Naive Bayes as a Generative Classifier

- Naive Bayes learns the *joint probability distribution* $p(\mathbf{x}, C_k) = P(\mathbf{x}|C_k)p(C_k)$.

Naive Bayes as a Generative Classifier

- Naive Bayes learns the *joint probability distribution* $p(\mathbf{x}, C_k) = P(\mathbf{x}|C_k)p(C_k)$. Hence it is a *generative classifier.*

Naive Bayes as a Generative Classifier

- Naive Bayes learns the *joint probability distribution* $p(\mathbf{x}, C_k) = P(\mathbf{x}|C_k)p(C_k)$. Hence it is a *generative classifier.*
- When we have the distribution $p(\mathbf{x}, C)$, we can sample features of elements belonging to each class C.

Naive Bayes as a Generative Classifier

- Naive Bayes learns the *joint probability distribution* $p(\mathbf{x}, C_k) = P(\mathbf{x}|C_k)p(C_k)$. Hence it is a *generative classifier.*
- When we have the distribution $p(\mathbf{x}, C)$, we can sample features of elements belonging to each class C.
- For example in Naive Bayes with Bernoulli distribution, we are modeling which words would occur in each class.

Naive Bayes as a Generative Classifier

- Naive Bayes learns the *joint probability distribution* $p(\mathbf{x}, C_k) = P(\mathbf{x}|C_k)p(C_k)$. Hence it is a *generative classifier.*
- When we have the distribution $p(\mathbf{x}, C)$, we can sample features of elements belonging to each class C.
- For example in Naive Bayes with Bernoulli distribution, we are modeling which words would occur in each class. A short text can, to some extent, be reconsructed from this knowledge.

Naive Bayes as a Generative Classifier

- Naive Bayes learns the *joint probability distribution* $p(\mathbf{x}, C_k) = P(\mathbf{x}|C_k)p(C_k)$. Hence it is a *generative classifier.*
- When we have the distribution $p(\mathbf{x}, C)$, we can sample features of elements belonging to each class C.
- For example in Naive Bayes with Bernoulli distribution, we are modeling which words would occur in each class. A short text can, to some extent, be reconsructed from this knowledge.
- Generative models model how each class "generates" data.

Naive Bayes as a Generative Classifier

- Naive Bayes learns the *joint probability distribution* $p(\mathbf{x}, C_k) = P(\mathbf{x}|C_k)p(C_k)$. Hence it is a *generative classifier.*
- When we have the distribution $p(\mathbf{x}, C)$, we can sample features of elements belonging to each class C.
- For example in Naive Bayes with Bernoulli distribution, we are modeling which words would occur in each class. A short text can, to some extent, be reconsructed from this knowledge.
- Generative models model how each class "generates" data.
- Discriminative models, such as Logistic Regression, directly learn $p(C|\mathbf{x})$, i.e. which features are most useful in distinguishing between classes.

Naive Bayes as a Generative Classifier

- Naive Bayes learns the *joint probability distribution* $p(\mathbf{x}, C_k) = P(\mathbf{x}|C_k)p(C_k)$. Hence it is a *generative classifier.*
- When we have the distribution $p(\mathbf{x}, C)$, we can sample features of elements belonging to each class C.
- For example in Naive Bayes with Bernoulli distribution, we are modeling which words would occur in each class. A short text can, to some extent, be reconsructed from this knowledge.
- Generative models model how each class "generates" data.
- Discriminative models, such as Logistic Regression, directly learn $p(C|\mathbf{x})$, i.e. which features are most useful in distinguishing between classes.
- Discriminative models cannot tell whether an instance is likely or not.

Naive Bayes as a Generative Classifier, cont.

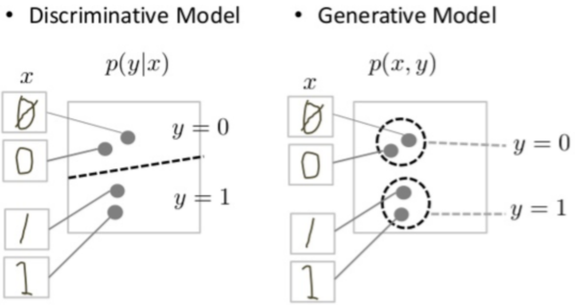

Figure: Credit: Google ML