Generative Learning Models

Dr. Víctor Uc Cetina

Facultad de Matemáticas
Universidad Autónoma de Yucatán

cetina@informatik.uni-hamburg.de
https://sites.google.com/view/victoruccetina

Content

Discriminative Vs Generative

- Consider a classification problem in which we want to distinguish between elephants ($y = 1$) and dogs ($y = 0$), based on some features of an animal.

Discriminative Vs Generative

- Consider a classification problem in which we want to distinguish between elephants ($y = 1$) and dogs ($y = 0$), based on some features of an animal.
- Given a training set, using an algorithm such as logistic regression or perceptron, we would find a line that works as a decision boundary that separates elephants and dogs.

Discriminative Vs Generative

- Consider a classification problem in which we want to distinguish between elephants ($y = 1$) and dogs ($y = 0$), based on some features of an animal.
- Given a training set, using an algorithm such as logistic regression or perceptron, we would find a line that works as a decision boundary that separates elephants and dogs.
- With generative models the approach is different.

Discriminative Vs Generative

- Consider a classification problem in which we want to distinguish between elephants ($y = 1$) and dogs ($y = 0$), based on some features of an animal.
- Given a training set, using an algorithm such as logistic regression or perceptron, we would find a line that works as a decision boundary that separates elephants and dogs.
- With generative models the approach is different.
- First, we build a model of how elephants look like; then we build a separate model of how dogs look like. Finally to classify a new animal we just need to match the animal with the elephant model, and then match it against the dog model.

Elements of a Generative Model

$y = 0$ indicates that the example is a dog

Elements of a Generative Model

$y = 0$ indicates that the example is a dog

$y = 1$ indicates that the example is an elephant

Elements of a Generative Model

$y = 0$ indicates that the example is a dog

$y = 1$ indicates that the example is an elephant

$p(x|y = 0)$ models the distribution of dogs' features

Elements of a Generative Model

$y = 0$ indicates that the example is a dog

$y = 1$ indicates that the example is an elephant

$p(x|y = 0)$ models the distribution of dogs' features

$p(x|y = 1)$ models the distribution of elephants' features

Elements of a Generative Model

$y = 0$ indicates that the example is a dog

$y = 1$ indicates that the example is an elephant

$p(x|y = 0)$ models the distribution of dogs' features

$p(x|y = 1)$ models the distribution of elephants' features

After modeling $p(y)$ and $p(x|y)$ we can use Bayes rule to derive the posterior distribution on y given x

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

Elements of a Generative Model

$y = 0$ indicates that the example is a dog

$y = 1$ indicates that the example is an elephant

$p(x|y = 0)$ models the distribution of dogs' features

$p(x|y = 1)$ models the distribution of elephants' features

After modeling $p(y)$ and $p(x|y)$ we can use Bayes rule to derive the posterior distribution on y given x

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

where

$$p(x) = p(x|y = 0)p(y = 0) + p(x|y = 1)p(y = 1)$$

Making a Prediction

If we were calculating $p(y|x)$ to make a prediction, we don't actually need to calculate the denominator:

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

Making a Prediction

If we were calculating $p(y|x)$ to make a prediction, we don't actually need to calculate the denominator:

$$\begin{aligned} \arg\max_y p(y|x) &= \arg\max_y \frac{p(x|y)p(y)}{p(x)} \\ &= \arg\max_y p(x|y)p(y). \end{aligned}$$

Multivariate Normal Distribution

In this model we assume that $p(x|y)$ is distributed according to a multivariate normal distribution in n-dimensions.

$$p(x;\mu,\mathbf{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^\top\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right)$$

where

$$\mu \in \mathbb{R}^n \text{ is a mean vector}$$

and

$$\mathbf{\Sigma} \in \mathbb{R}^{n\times n} \text{ is a covariance matrix and } \mathbf{\Sigma} \geq 0$$

GDA Model

Suppose a classification problem in which the input features x are continuous-valued random variables, we can use a GDA model:

$$y \sim \texttt{Bernoulli}(\phi)$$

GDA Model

Suppose a classification problem in which the input features x are continuous-valued random variables, we can use a GDA model:

$$\begin{aligned} y &\sim \texttt{Bernoulli}(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \end{aligned}$$

GDA Model

Suppose a classification problem in which the input features x are continuous-valued random variables, we can use a GDA model:

$$\begin{aligned} y &\sim \texttt{Bernoulli}(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu_1, \Sigma) \end{aligned}$$

GDA Model

Suppose a classification problem in which the input features x are continuous-valued random variables, we can use a GDA model:

$$\begin{aligned} y &\sim \texttt{Bernoulli}(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu_1, \Sigma) \end{aligned}$$

Writing out the distribution this is:

$$p(y) = \phi^y (1-\phi)^{1-y}$$

GDA Model

Suppose a classification problem in which the input features x are continuous-valued random variables, we can use a GDA model:

$$\begin{aligned} y &\sim \texttt{Bernoulli}(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu_1, \Sigma) \end{aligned}$$

Writing out the distribution this is:

$$\begin{aligned} p(y) &= \phi^y(1-\phi)^{1-y} \\ p(x|y=0) &= \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}}\texttt{exp}\left(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{0})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu_\mathbf{0})\right) \end{aligned}$$

GDA Model

Suppose a classification problem in which the input features x are continuous-valued random variables, we can use a GDA model:

$$\begin{aligned} y &\sim \texttt{Bernoulli}(\phi) \\ x|y=0 &\sim \mathcal{N}(\mu_0, \Sigma) \\ x|y=1 &\sim \mathcal{N}(\mu_1, \Sigma) \end{aligned}$$

Writing out the distribution this is:

$$\begin{aligned} p(y) &= \phi^y(1-\phi)^{1-y} \\ p(x|y=0) &= \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{0})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu_\mathbf{0})\right) \\ p(x|y=1) &= \frac{1}{(2\pi)^{n/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{1})^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu_\mathbf{1})\right) \end{aligned}$$

Log-Likelihood

The log-likelihood of the data is given by

$$\ell(\phi, \mu_{\mathbf{0}}, \mu_{\mathbf{1}}, \mathbf{\Sigma}) \quad = \quad \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}; \phi, \mu_{\mathbf{0}}, \mu_{\mathbf{1}}, \mathbf{\Sigma})$$

Log-Likelihood

The log-likelihood of the data is given by

$$\begin{aligned}\ell(\phi, \mu_{\mathbf{0}}, \mu_{\mathbf{1}}, \mathbf{\Sigma}) &= \log \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}; \phi, \mu_{\mathbf{0}}, \mu_{\mathbf{1}}, \mathbf{\Sigma}) \\ &= \log \prod_{i=1}^{m} p(x^{(i)} | y^{(i)}; \phi, \mu_{\mathbf{0}}, \mu_{\mathbf{1}}, \mathbf{\Sigma}) p(y^{(i)}; \phi).\end{aligned}$$

Maximizing ℓ

By maximizing ℓ with respect to the parameters we find the maximum likelihood estimate of the parameters to be:

Maximizing ℓ

By maximizing ℓ with respect to the parameters we find the maximum likelihood estimate of the parameters to be:

$$\phi = \frac{1}{m}\sum_{i=1}^{m} 1\{y^{(i)} = 1\}$$

Maximizing ℓ

By maximizing ℓ with respect to the parameters we find the maximum likelihood estimate of the parameters to be:

$$\phi = \frac{1}{m}\sum_{i=1}^{m} 1\{y^{(i)} = 1\}$$

$$\mu_{\mathbf{0}} = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=0\}x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}}$$

Maximizing ℓ

By maximizing ℓ with respect to the parameters we find the maximum likelihood estimate of the parameters to be:

$$\phi = \frac{1}{m}\sum_{i=1}^{m} 1\{y^{(i)} = 1\}$$

$$\mu_{\mathbf{0}} = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=0\}x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}}$$

$$\mu_{\mathbf{1}} = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}$$

Maximizing ℓ

By maximizing ℓ with respect to the parameters we find the maximum likelihood estimate of the parameters to be:

$$
\begin{aligned}
\phi &= \tfrac{1}{m} \textstyle\sum_{i=1}^{m} 1\{y^{(i)} = 1\} \\
\boldsymbol{\mu_0} &= \frac{\sum_{i=1}^{m} 1\{y^{(i)}=0\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}} \\
\boldsymbol{\mu_1} &= \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\} x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}} \\
\boldsymbol{\Sigma} &= \tfrac{1}{m} \textstyle\sum_{i=1}^{m} (x^{(i)} - \mu_{y^{(i)}})(x^{(i)} - \mu_{y^{(i)}})^{\top}
\end{aligned}
$$

GDA and Logistic Regression

- The GDA model has an interesting relationship to logistic regression.

GDA and Logistic Regression

- The GDA model has an interesting relationship to logistic regression.
- If we view the quantity $p(y = 1|\mathbf{x}; \phi, \mu_{\mathbf{0}}, \mu_{\mathbf{1}}, \mathbf{\Sigma})$ as a function of $\mathbf{x}$, we'll find that it can be expressed in the form

$$p(y = 1|\mathbf{x}; \phi, \mu_{\mathbf{0}}, \mu_{\mathbf{1}}, \mathbf{\Sigma}) = \frac{1}{1 + \exp(-\theta^{\top}\mathbf{x})}$$

where θ is some appropriate function of $\phi, \mu_{\mathbf{0}}, \mu_{\mathbf{1}}, \mathbf{\Sigma}$. This is exactly the form that logistic regression used to model $p(y = 1|\mathbf{x})$.

GDA Vs Logistic Regression

- If $p(\mathbf{x}|y)$ is a multivariate Gaussian then $p(y|\mathbf{x})$ necessarily follows a logistic function.

GDA Vs Logistic Regression

- If $p(\mathbf{x}|y)$ is a multivariate Gaussian then $p(y|\mathbf{x})$ necessarily follows a logistic function.
- The converse however, is not true. Being $p(\mathbf{x}|y)$ a logistic function does not imply that $p(y|\mathbf{x})$ is a multivariate Gaussian.

GDA Vs Logistic Regression

- If $p(\mathbf{x}|y)$ is a multivariate Gaussian then $p(y|\mathbf{x})$ necessarily follows a logistic function.
- The converse however, is not true. Being $p(\mathbf{x}|y)$ a logistic function does not imply that $p(y|\mathbf{x})$ is a multivariate Gaussian.
- GDA makes stronger modeling assumptions, and is more data efficient (i.e. it requires less training data to learn well) when the modeling assumptions are correct or at least approximately correct.

GDA Vs Logistic Regression

- If $p(\mathbf{x}|y)$ is a multivariate Gaussian then $p(y|\mathbf{x})$ necessarily follows a logistic function.
- The converse however, is not true. Being $p(\mathbf{x}|y)$ a logistic function does not imply that $p(y|\mathbf{x})$ is a multivariate Gaussian.
- GDA makes stronger modeling assumptions, and is more data efficient (i.e. it requires less training data to learn well) when the modeling assumptions are correct or at least approximately correct.
- Logistic regression makes weaker assumptions, and is significantly more robust to deviations from modeling assumptions.

GDA Vs Logistic Regression

- If $p(\mathbf{x}|y)$ is a multivariate Gaussian then $p(y|\mathbf{x})$ necessarily follows a logistic function.
- The converse however, is not true. Being $p(\mathbf{x}|y)$ a logistic function does not imply that $p(y|\mathbf{x})$ is a multivariate Gaussian.
- GDA makes stronger modeling assumptions, and is more data efficient (i.e. it requires less training data to learn well) when the modeling assumptions are correct or at least approximately correct.
- Logistic regression makes weaker assumptions, and is significantly more robust to deviations from modeling assumptions.
- Specifically, when the data is non-Gaussian, then in the limit of large datasets, logistic regression will almost always do better than GDA.

Naive Bayes

In GDA the features vectors $\mathbf{x}$ were continuous, real-valued vectors. Lets talk about a different learning algorithm for discrete-valued vectors.

Naive Bayes

In GDA the features vectors $\mathbf{x}$ were continuous, real-valued vectors. Lets talk about a different learning algorithm for discrete-valued vectors.

- Consider building an email spam filter using machine learning.

Naive Bayes

In GDA the features vectors $\mathbf{x}$ were continuous, real-valued vectors. Lets talk about a different learning algorithm for discrete-valued vectors.

- Consider building an email spam filter using machine learning.
- Lets say we have a training set: a set of emails labeled as spam or non-spam.

Naive Bayes

In GDA the features vectors $\mathbf{x}$ were continuous, real-valued vectors. Lets talk about a different learning algorithm for discrete-valued vectors.

- Consider building an email spam filter using machine learning.
- Lets say we have a training set: a set of emails labeled as spam or non-spam.
- We'll begin the construction of our spam filter by specifying the features x_i used to represent an email.

Naive Bayes

In GDA the features vectors **x** were continuous, real-valued vectors. Lets talk about a different learning algorithm for discrete-valued vectors.

- Consider building an email spam filter using machine learning.
- Lets say we have a training set: a set of emails labeled as spam or non-spam.
- We'll begin the construction of our spam filter by specifying the features x_i used to represent an email.
- We will represent an email via a feature vector whose length is equal to the number of words in a dictionary.

Naive Bayes

In GDA the features vectors $\mathbf{x}$ were continuous, real-valued vectors. Lets talk about a different learning algorithm for discrete-valued vectors.

- Consider building an email spam filter using machine learning.
- Lets say we have a training set: a set of emails labeled as spam or non-spam.
- We'll begin the construction of our spam filter by specifying the features x_i used to represent an email.
- We will represent an email via a feature vector whose length is equal to the number of words in a dictionary.
- Specifically, if an email contains the i-th word of the dictionary, then we will set $x_i = 1$; otherwise, we let $x_i = 0$.

Features Vector

The set of words encoded into de feature vector is called the vocabulary, so the dimension of **x** is equal to the size of the vocabulary.

$$\mathbf{x} = \begin{array}{cl} 1 & \text{a} \\ 0 & \text{after} \\ \vdots & \vdots \\ 1 & \text{buy} \\ \vdots & \vdots \\ 0 & \text{zebra} \end{array}$$

Naive Bayes Assumption

- Having chosen our feature vector, we now want to build a discriminative model. So we have to model $p(\mathbf{x}|y)$.

Naive Bayes Assumption

- Having chosen our feature vector, we now want to build a discriminative model. So we have to model $p(\mathbf{x}|y)$.
- With a vocabulary of 50000 words, then $\mathbf{x} \in \{0, 1\}^{50000}$ and if we were to model $\mathbf{x}$ explicitly with a multinomial distribution over the 2^{50000} possible outcomes, then we would end up with a $(2^{50000} - 1)$-dimensional parameter vector. This is clearly too many parameters.

Naive Bayes Assumption

- Having chosen our feature vector, we now want to build a discriminative model. So we have to model $p(\mathbf{x}|y)$.
- With a vocabulary of 50000 words, then $\mathbf{x} \in \{0, 1\}^{50000}$ and if we were to model $\mathbf{x}$ explicitly with a multinomial distribution over the 2^{50000} possible outcomes, then we would end up with a $(2^{50000} - 1)$-dimensional parameter vector. This is clearly too many parameters.
- To model $p(\mathbf{x}|y)$ we will therefore make a very strong assumption. We will assume that the $x_i's$ are conditionally independent given y.

Naive Bayes Assumption

- Having chosen our feature vector, we now want to build a discriminative model. So we have to model $p(\mathbf{x}|y)$.
- With a vocabulary of 50000 words, then $\mathbf{x} \in \{0,1\}^{50000}$ and if we were to model $\mathbf{x}$ explicitly with a multinomial distribution over the 2^{50000} possible outcomes, then we would end up with a $(2^{50000} - 1)$-dimensional parameter vector. This is clearly too many parameters.
- To model $p(\mathbf{x}|y)$ we will therefore make a very strong assumption. We will assume that the $x_i's$ are conditionally independent given y.
- This assumption is called the Naive Bayes Assumption and the resulting algorithm is called the Naive Bayes Classifier.

Naive Bayes Classifier

- For instance, if $y = 1$ means spam email; “buy” is word 2087 and “price” is word 39831; then we are assuming that if I tell you $y = 1$, then knowledge of x_{2087} will have no effect on your beliefs about the value of x_{39831}.

Naive Bayes Classifier

- For instance, if $y = 1$ means spam email; “buy” is word 2087 and “price” is word 39831; then we are assuming that if I tell you $y = 1$, then knowledge of x_{2087} will have no effect on your beliefs about the value of x_{39831}.
- More formally $p(x_{2087}|x_{39831}, y) = p(x_{2087}|y)$.

Naive Bayes Classifier

- For instance, if $y = 1$ means spam email; “buy” is word 2087 and “price” is word 39831; then we are assuming that if I tell you $y = 1$, then knowledge of x_{2087} will have no effect on your beliefs about the value of x_{39831}.
- More formally $p(x_{2087}|x_{39831}, y) = p(x_{2087}|y)$.
- This is not the same as saying that x_{2087} and x_{39831} are independent, which would have been written $p(x_{2087}) = p(x_{2087}|x_{39831})$.

Naive Bayes Classifier

$$p(x_1, \ldots, x_{50000} | y)$$

Naive Bayes Classifier

$$
\begin{aligned}
& p(x_1, \ldots, x_{50000} | y) \\
= \quad & p(x_1|y)p(x_2|y, x_1)p(x_3|y, x_1, x_2) \cdots p(x_{50000}|y, x_1, \ldots, x_{49999})
\end{aligned}
$$

Naive Bayes Classifier

$$
\begin{aligned}
& p(x_1, \ldots, x_{50000} | y) \\
& \quad = \quad p(x_1|y)p(x_2|y, x_1)p(x_3|y, x_1, x_2) \cdots p(x_{50000}|y, x_1, \ldots, x_{49999}) \\
& \quad = \quad p(x_1|y)p(x_2|y)p(x_3|y) \cdots p(x_{50000}|y)
\end{aligned}
$$

Naive Bayes Classifier

$$
\begin{aligned}
& p(x_1, \ldots, x_{50000} | y) \\
& \quad = \quad p(x_1|y)p(x_2|y, x_1)p(x_3|y, x_1, x_2) \cdots p(x_{50000}|y, x_1, \ldots, x_{49999}) \\
& \quad = \quad p(x_1|y)p(x_2|y)p(x_3|y) \cdots p(x_{50000}|y) \\
& \quad = \quad \prod_{i=1}^{n} p(x_i|y)
\end{aligned}
$$

Parameters of the Model

Our model is parameterized by

Parameters of the Model

Our model is parameterized by

$$\phi_{i|y=1} = p(x_i = 1|y = 1)$$

Parameters of the Model

Our model is parameterized by

$\phi_{i|y=1} = p(x_i = 1|y = 1)$

$\phi_{i|y=0} = p(x_i = 1|y = 0)$, and

Parameters of the Model

Our model is parameterized by

$\phi_{i|y=1} = p(x_i = 1|y = 1)$

$\phi_{i|y=0} = p(x_i = 1|y = 0)$, and

$\phi_y = p(y = 1)$

Parameters of the Model

Our model is parameterized by

$$\phi_{i|y=1} = p(x_i = 1|y = 1)$$

$$\phi_{i|y=0} = p(x_i = 1|y = 0), \text{ and}$$

$$\phi_y = p(y = 1)$$

Given a training set $\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ we can write down the joint likelihood of the data:

$$L(\phi_y, \phi_{i|y=0}, \phi_{i|y=1}) = \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}).$$

Maximum Likelihood Estimates

Maximizing $L(\phi_y, \phi_{i|y=0}, \phi_{i|y=1})$ with respect to $\phi_y, \phi_{i|y=0}, \phi_{i|y=1}$ gives the maximum likelihood estimates:

Maximum Likelihood Estimates

Maximizing $L(\phi_y, \phi_{i|y=0}, \phi_{i|y=1})$ with respect to $\phi_y, \phi_{i|y=0}, \phi_{i|y=1}$ gives the maximum likelihood estimates:

Fraction of the spam emails in which word j does appear:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=1\}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}$$

Maximum Likelihood Estimates

Maximizing $L(\phi_y, \phi_{i|y=0}, \phi_{i|y=1})$ with respect to $\phi_y, \phi_{i|y=0}, \phi_{i|y=1}$ gives the maximum likelihood estimates:

Fraction of the spam emails in which word j does appear:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=1\}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}$$

Fraction of the non-spam emails in which word j does appear:

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=0\}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}}$$

Maximum Likelihood Estimates

Maximizing $L(\phi_y, \phi_{i|y=0}, \phi_{i|y=1})$ with respect to $\phi_y, \phi_{i|y=0}, \phi_{i|y=1}$ gives the maximum likelihood estimates:

Fraction of the spam emails in which word j does appear:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=1\}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}$$

Fraction of the non-spam emails in which word j does appear:

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=0\}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}}$$

Fraction of emails which are spam emails

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}{m}$$

Making Predictions

To make a prediction on a new example with features $\mathbf{x}$, we simply calculate:

$$p(y = 1|\mathbf{x}) \quad = \quad \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x})}$$

Making Predictions

To make a prediction on a new example with features **x**, we simply calculate:

$$\begin{aligned} p(y=1|\mathbf{x}) &= \frac{p(\mathbf{x}|y{=}1)p(y{=}1)}{p(\mathbf{x})} \\ &= \frac{(\prod_{i=1}^{n} p(x_i|y{=}1))p(y{=}1)}{(\prod_{i=1}^{n} p(x_i|y{=}1))p(y{=}1)+(\prod_{i=1}^{n} p(x_i|y{=}0))p(y{=}0)} \end{aligned}$$

where n is the size of the vocabulary.

Naive Bayes

- While we have developed the Naive Bayes algorithm mainly for the case of problems where the features x_i are binary-valued, the generalization to where x_i can take values in $\{1, 2, \ldots, k_i\}$ is straightforward.

Naive Bayes

- While we have developed the Naive Bayes algorithm mainly for the case of problems where the features x_i are binary-valued, the generalization to where x_i can take values in $\{1, 2, \ldots, k_i\}$ is straightforward.
- We would simply model $p(x_i|y)$ as multinomial rather than as Bernoulli.

Naive Bayes

- Even if some original input attribute (say, the living area of a house, as in our earlier example) were continuous valued, it is quite common to discretize it and apply Naive Bayes.

Naive Bayes

- Even if some original input attribute (say, the living area of a house, as in our earlier example) were continuous valued, it is quite common to discretize it and apply Naive Bayes.
- For instance, if we use some feature x_i to represent living area, we might discretize the continuous values as follows:

Living area	< 400	$400 - 800$	$800 - 1200$	> 1200
x_i	1	2	3	4

Naive Bayes

- Even if some original input attribute (say, the living area of a house, as in our earlier example) were continuous valued, it is quite common to discretize it and apply Naive Bayes.
- For instance, if we use some feature x_i to represent living area, we might discretize the continuous values as follows:

Living area	< 400	$400 - 800$	$800 - 1200$	> 1200
x_i	1	2	3	4

- Thus, for a house with living area 890 square feet, we would set the value of the corresponding feature x_i to 3.

Laplace Smoothing

- The Naive Bayes algorithm as we have described it will work fairly well for many problems, but there is a simple change that makes it work much better, especially for text classification.

Laplace Smoothing

- The Naive Bayes algorithm as we have described it will work fairly well for many problems, but there is a simple change that makes it work much better, especially for text classification.
- It is statistically a bad idea to estimate the probability of some event to be zero just because you haven't seen it before in your finite training set. This is exactly what it happens when a new word appears in a new email.

Laplace Smoothing

- The Naive Bayes algorithm as we have described it will work fairly well for many problems, but there is a simple change that makes it work much better, especially for text classification.
- It is statistically a bad idea to estimate the probability of some event to be zero just because you haven't seen it before in your finite training set. This is exactly what it happens when a new word appears in a new email.
- Lets assume that the word 35000th in the dictionary is "nips" and this word never occurs in any of the emails of our training set.

Laplace Smoothing

- Then, its probabilities would be computed as follows:

Laplace Smoothing

- Then, its probabilities would be computed as follows:
- $\phi_{35000|y=1} = \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)}=1, y^{(i)}=1\}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}} = 0$

Laplace Smoothing

- Then, its probabilities would be computed as follows:
- $\phi_{35000|y=1} = \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)}=1, y^{(i)}=1\}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}} = 0$
- $\phi_{35000|y=0} = \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)}=1, y^{(i)}=0\}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}} = 0$

Laplace Smoothing

- Then, its probabilities would be computed as follows:
- $\phi_{35000|y=1} = \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)}=1, y^{(i)}=1\}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}} = 0$
- $\phi_{35000|y=0} = \frac{\sum_{i=1}^{m} 1\{x_{35000}^{(i)}=1, y^{(i)}=0\}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}} = 0$
- Hence, when trying to decide if one of these messages containing "nips" is spam, it calculates the class posterior probabilities, and obtains:

$$p(y=1|x) = \frac{\prod_{i=1}^{n} p(x_i|y=1)p(y=1)}{\prod_{i=1}^{n} p(x_i|y=1)p(y=1) + \prod_{i=1}^{n} p(x_i|y=0)p(y=0)} = \frac{0}{0}$$

since there is always a term $p(x_{35000}|y) = 0$ in $\prod_{i=1}^{n} p(x_i|y)$.

Laplace Smoothing

- Take the problem of estimating the mean of a multinomial random variable z taking values in $\{1, \ldots, k\}$.

Laplace Smoothing

- Take the problem of estimating the mean of a multinomial random variable z taking values in $\{1, \ldots, k\}$.
- We can parameterize our multinomial with $\phi_i = p(z = i)$.

Laplace Smoothing

- Take the problem of estimating the mean of a multinomial random variable z taking values in $\{1, \ldots, k\}$.
- We can parameterize our multinomial with $\phi_i = p(z = i)$.
- Therefore, given a set of m independent observations $\{z^{(1)}, \ldots, z^{(m)}\}$ the maximum likelihood estimates are given by

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}{m}.$$

Laplace Smoothing

So, how can we fix it?

- As we saw previously, if we were to use these maximum likelihood estimates, then some of the ϕ_j's might end up as zero, which was a problem. To avoid this, we can use Laplace smoothing, which replaces (1) with (2).

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}{m}. \tag{1}$$

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} + 1}{m + k}. \tag{2}$$

Laplace Smoothing

So, how can we fix it?

- As we saw previously, if we were to use these maximum likelihood estimates, then some of the ϕ_j's might end up as zero, which was a problem. To avoid this, we can use Laplace smoothing, which replaces (1) with (2).

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}{m}. \tag{1}$$

$$\phi_j = \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} + 1}{m + k}. \tag{2}$$

- Here, we've added 1 to the numerator, and k to the denominator. $\sum_{j=1}^{k} \phi_j = 1$ still holds and now $\phi_j \neq 0$ for all values of j.

Laplace Smoothing

- Returning to our Naive Bayes classifier, with Laplace smoothing, we therefore obtain the following estimates of the parameters:

Laplace Smoothing

- Returning to our Naive Bayes classifier, with Laplace smoothing, we therefore obtain the following estimates of the parameters:
- $\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=1\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}+2}$

Laplace Smoothing

- Returning to our Naive Bayes classifier, with Laplace smoothing, we therefore obtain the following estimates of the parameters:
- $\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=1\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}+2}$
- $\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=0\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}+2}$

Laplace Smoothing

- Returning to our Naive Bayes classifier, with Laplace smoothing, we therefore obtain the following estimates of the parameters:
- $\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=1\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=1\}+2}$
- $\phi_{j|y=0} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)}=1, y^{(i)}=0\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=0\}+2}$
- In general this is called a multi-variate Bernoulli event model.

Multinomial Event Model

- To describe this model, we will use a different notation and set of features for representing emails. We let x_i denote the identity of the i-th word in the email. Thus, x_i is now an integer taking values in $\{1, ..., |V|\}$, where $|V|$ is the size of our vocabulary (dictionary).

Multinomial Event Model

- To describe this model, we will use a different notation and set of features for representing emails. We let x_i denote the identity of the i-th word in the email. Thus, x_i is now an integer taking values in $\{1, ..., |V|\}$, where $|V|$ is the size of our vocabulary (dictionary).
- An email of n words is now represented by a vector $(x_1, x_2, ..., x_n)$ of length n; note that n can vary for different documents. For instance, if an email starts with "A NIPS . . . ", then $x_1 = 1$ (if "A" is the first word in the dictionary), and $x_2 = 35000$ (if "nips" is the 35000th word in the dictionary).

Multinomial Event Model

- The parameters for our new model are $\phi_y = p(y)$ as before, $\phi_{i|y=1} = p(x_j = i|y = 1)$ (for any j) and $\phi_{i|y=0} = p(x_j = i|y = 0)$.

Multinomial Event Model

- The parameters for our new model are $\phi_y = p(y)$ as before, $\phi_{i|y=1} = p(x_j = i|y = 1)$ (for any j) and $\phi_{i|y=0} = p(x_j = i|y = 0)$.
- Note that we have assumed that $p(x_j|y)$ is the same for all values of j (i.e., that the distribution according to which a word is generated does not depend on its position j within the email).

Multinomial Event Model

If we are given a training set $\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ where $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_{n_i}^{(i)})$ (here, n_i is the number of words in the i-training example), the likelihood of the data is given by

$$\begin{aligned} L(\phi, \phi_{i|y=0}, \phi_{i|y=1}) &= \prod_{i=1}^{m} p(x^{(i)}, y^{(i)}) \\ & \prod_{i=1}^{m} \left(\prod_{j=1}^{n_i} p(x_j^{(i)}|y; \phi_{i|y=0}, \phi_{i|y=1}) \right) p(y^{(i)}; \phi_y) \end{aligned}$$

Multinomial Event Model

Maximizing this we obtain the maximum likelihood estimates of the parameters:

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)}=k, y=1\}}{\sum_{i=1}^{m} 1\{y^{(i)}=1\} n_i}$$

$$\phi_{k|y=0} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)}=k, y=0\}}{\sum_{i=1}^{m} 1\{y^{(i)}=0\} n_i}$$

$$\phi_y = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=1\}}{m}.$$

Multinomial Event Model

If we were to apply Laplace smoothing we obtain:

$$\phi_{k|y=1} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)}=k, y=1\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=1\} n_i + |V|}$$

$$\phi_{k|y=0} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)}=k, y=0\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=0\} n_i + |V|}$$

where $|V|$ is the size of the vocabulary.

Decision Theory

- Suppose we have an input vector $\mathbf{x}$ together with a corresponding vector $\mathbf{t}$ of target variables, and our goal is to predict $\mathbf{t}$ given a new value for $\mathbf{x}$.

Decision Theory

- Suppose we have an input vector $\mathbf{x}$ together with a corresponding vector $\mathbf{t}$ of target variables, and our goal is to predict $\mathbf{t}$ given a new value for $\mathbf{x}$.
- For regression problems, $\mathbf{t}$ will comprise continuous variables, whereas for classification problems $\mathbf{t}$ will represent class labels.

Decision Theory

- Suppose we have an input vector $\mathbf{x}$ together with a corresponding vector $\mathbf{t}$ of target variables, and our goal is to predict $\mathbf{t}$ given a new value for $\mathbf{x}$.
- For regression problems, $\mathbf{t}$ will comprise continuous variables, whereas for classification problems $\mathbf{t}$ will represent class labels.
- The joint probability distribution $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of the uncertainty associated with these variables.

Decision Theory

- Suppose we have an input vector $\mathbf{x}$ together with a corresponding vector $\mathbf{t}$ of target variables, and our goal is to predict $\mathbf{t}$ given a new value for $\mathbf{x}$.
- For regression problems, $\mathbf{t}$ will comprise continuous variables, whereas for classification problems $\mathbf{t}$ will represent class labels.
- The joint probability distribution $p(\mathbf{x}, \mathbf{t})$ provides a complete summary of the uncertainty associated with these variables.
- Determination of $p(\mathbf{x}, \mathbf{t})$ from a set of training data is an example of inference and is typically a very difficult problem.

Decision Theory

- In a practical application, however, we must often make a specific prediction for the value of $\mathbf{t}$, or more generally take a specific action based on our understanding of the values $\mathbf{t}$ is likely to take, and this aspect is the subject of decision theory.

Decision Theory

- Consider, for example, a medical diagnosis problem in which we have taken an X-ray image of a patient, and we wish to determine whether the patient has cancer or not.

Decision Theory

- Consider, for example, a medical diagnosis problem in which we have taken an X-ray image of a patient, and we wish to determine whether the patient has cancer or not.
- In this case, the input vector $\mathbf{x}$ is the set of pixel intensities in the image,

Decision Theory

- Consider, for example, a medical diagnosis problem in which we have taken an X-ray image of a patient, and we wish to determine whether the patient has cancer or not.
- In this case, the input vector $\mathbf{x}$ is the set of pixel intensities in the image,
- and output variable $\mathbf{t}$ will represent the presence of cancer,

Decision Theory

- Consider, for example, a medical diagnosis problem in which we have taken an X-ray image of a patient, and we wish to determine whether the patient has cancer or not.
- In this case, the input vector $\mathbf{x}$ is the set of pixel intensities in the image,
- and output variable $\mathbf{t}$ will represent the presence of cancer,
- which we denote by the class C_1, or the absence of cancer, which we denote by the class C_2.

Decision Theory

- We might, for instance, choose $\mathbf{t}$ to be a binary variable such that $\mathbf{t} = 0$ corresponds to class C_1 and $\mathbf{t} = 1$ corresponds to class C_2.

Decision Theory

- We might, for instance, choose $\mathbf{t}$ to be a binary variable such that $\mathbf{t} = 0$ corresponds to class C_1 and $\mathbf{t} = 1$ corresponds to class C_2.
- The general inference problem then involves determining the joint distribution $p(\mathbf{x}, C_k)$, or equivalently $p(\mathbf{x}, \mathbf{t})$, which gives us the most complete probabilistic description of the situation.

Decision Theory

- We might, for instance, choose $\mathbf{t}$ to be a binary variable such that $\mathbf{t} = 0$ corresponds to class C_1 and $\mathbf{t} = 1$ corresponds to class C_2.
- The general inference problem then involves determining the joint distribution $p(\mathbf{x}, C_k)$, or equivalently $p(\mathbf{x}, \mathbf{t})$, which gives us the most complete probabilistic description of the situation.
- Although this can be a very useful and informative quantity, in the end we must decide either to give treatment to the patient or not, and we would like this choice to be optimal in some appropriate sense.

Decision Theory

- We might, for instance, choose $\mathbf{t}$ to be a binary variable such that $\mathbf{t} = 0$ corresponds to class C_1 and $\mathbf{t} = 1$ corresponds to class C_2.
- The general inference problem then involves determining the joint distribution $p(\mathbf{x}, C_k)$, or equivalently $p(\mathbf{x}, \mathbf{t})$, which gives us the most complete probabilistic description of the situation.
- Although this can be a very useful and informative quantity, in the end we must decide either to give treatment to the patient or not, and we would like this choice to be optimal in some appropriate sense.
- We shall see that the decision stage is generally very simple, even trivial, once we have solved the inference problem.

Decision Theory

- We are interested in the probabilities of the two classes given the image, which are given by $p(C_k|\mathbf{x})$. Using Bayes' theorem, these probabilities can be expressed in the form

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})}.$$

Decision Theory

- We are interested in the probabilities of the two classes given the image, which are given by $p(C_k|\mathbf{x})$. Using Bayes' theorem, these probabilities can be expressed in the form

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})}.$$

- Note that any of the quantities appearing in Bayes' theorem can be obtained from the joint distribution $p(\mathbf{x}, C_k)$ by either marginalizing or conditioning with respect to the appropriate variables.

Decision Theory

Minimizing the misclassification rate:

- Suppose that our goal is simply to make as few misclassifications as possible.

Decision Theory

Minimizing the misclassification rate:

- Suppose that our goal is simply to make as few misclassifications as possible.
- A mistake occurs when an input vector belonging to class C_1 is assigned to class C_2 or vice versa.

Decision Theory

Minimizing the misclassification rate:

- Suppose that our goal is simply to make as few misclassifications as possible.
- A mistake occurs when an input vector belonging to class C_1 is assigned to class C_2 or vice versa.
- A rule will divide the input space into regions R_k called decision regions, one for each class, such that all points in R_k are assigned to class C_k.

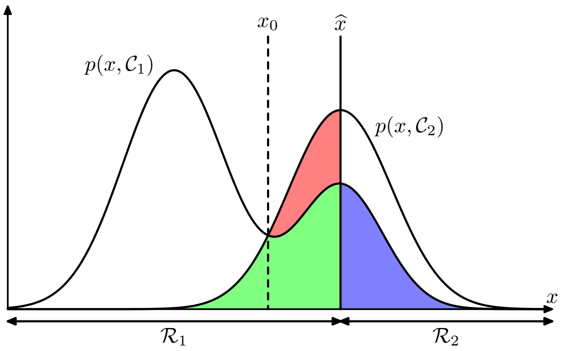
Decision Theory

Minimizing the misclassification rate:

- Suppose that our goal is simply to make as few misclassifications as possible.
- A mistake occurs when an input vector belonging to class C_1 is assigned to class C_2 or vice versa.
- A rule will divide the input space into regions R_k called decision regions, one for each class, such that all points in R_k are assigned to class C_k.
- The probability of this occurring is given by

$$p(\mathrm{mistake}) = p(\mathbf{x} \in R_1, C_2) + p(\mathbf{x} \in R_2, C_1)$$

Decision Theory



- If $p(\mathbf{x}, C_1) > p(\mathbf{x}, C_2)$ for a given value of $\mathbf{x}$, then we should assign that $\mathbf{x}$ to class C_1.

Decision Theory

- From the product rule of probability we have $p(\mathbf{x}, C_k) = p(C_k|\mathbf{x})p(\mathbf{x})$.

Decision Theory

- From the product rule of probability we have $p(\mathbf{x}, C_k) = p(C_k|\mathbf{x})p(\mathbf{x})$.
- Therefore, the minimum probability of making a mistake is obtained if each value of $\mathbf{x}$ is assigned to the class for which the posterior probability $p(C_k|\mathbf{x})$ is largest.

Decision Theory

Minimizing the expected loss:

- For many applications, our objective will be more complex than simply minimizing the number of misclassifications.

Decision Theory

Minimizing the expected loss:

- For many applications, our objective will be more complex than simply minimizing the number of misclassifications.
- Let us consider again the medical diagnosis problem. We note that, if a patient who does not have cancer is incorrectly diagnosed as having cancer, the consequences may be some patient distress plus the need for further investigations.

Decision Theory

Minimizing the expected loss:

- For many applications, our objective will be more complex than simply minimizing the number of misclassifications.
- Let us consider again the medical diagnosis problem. We note that, if a patient who does not have cancer is incorrectly diagnosed as having cancer, the consequences may be some patient distress plus the need for further investigations.
- Conversely, if a patient with cancer is diagnosed as healthy, the result may be premature death due to lack of treatment.

Decision Theory

- Thus the consequences of these two types of mistake can be dramatically different.

Decision Theory

- Thus the consequences of these two types of mistake can be dramatically different.
- It would clearly be better to make fewer mistakes of the second kind, even if this was at the expense of making more mistakes of the first kind.

Decision Theory

- Thus the consequences of these two types of mistake can be dramatically different.
- It would clearly be better to make fewer mistakes of the second kind, even if this was at the expense of making more mistakes of the first kind.
- We can formalize such issues through the introduction of a loss function, also called a cost function, which is a single, overall measure of loss incurred in taking any of the available decisions or actions.

Decision Theory

- Thus the consequences of these two types of mistake can be dramatically different.
- It would clearly be better to make fewer mistakes of the second kind, even if this was at the expense of making more mistakes of the first kind.
- We can formalize such issues through the introduction of a loss function, also called a cost function, which is a single, overall measure of loss incurred in taking any of the available decisions or actions.
- Our goal is then to minimize the total loss incurred.

Decision Theory

- Suppose that, for a new value of $\mathbf{x}$, the true class is C_k and that we assign $\mathbf{x}$ to class C_j (where j may or may not be equal to k).

Decision Theory

- Suppose that, for a new value of **x**, the true class is C_k and that we assign **x** to class C_j (where j may or may not be equal to k).
- In so doing, we incur some level of loss that we denote by L_{kj}, which we can view as the k, j element of a loss matrix.

	cancer	*normal*
cancer	0	1000
normal	1	0

Decision Theory

- Suppose that, for a new value of $\mathbf{x}$, the true class is C_k and that we assign $\mathbf{x}$ to class C_j (where j may or may not be equal to k).
- In so doing, we incur some level of loss that we denote by L_{kj}, which we can view as the k, j element of a loss matrix.

	cancer	*normal*
cancer	0	1000
normal	1	0

- Our goal is to choose the regions R_j in order to minimize the expected loss, which implies that for each $\mathbf{x}$ we should minimize $\sum_k L_{kj} p(\mathbf{x}, C_k)$.

Decision Theory

- As before, we can use the product rule $p(\mathbf{x}, C_k) = p(C_k|\mathbf{x})p(\mathbf{x})$ to eliminate the common factor of $p(\mathbf{x})$.

Decision Theory

- As before, we can use the product rule $p(\mathbf{x}, C_k) = p(C_k|\mathbf{x})p(\mathbf{x})$ to eliminate the common factor of $p(\mathbf{x})$.
- Thus the decision rule that minimizes the expected loss is the one that assigns each new $\mathbf{x}$ to the class j for which the quantity

$$\sum_k L_{kj} p(C_k|\mathbf{x})$$

  is a minimum.

Reference

- Andrew Ng. **Machine Learning Course Notes**. 2003.
- Christopher Bishop. **Pattern Recognition and Machine Learning**. Springer. 2006.

Thank you!

Dr. Victor Uc Cetina
cetina@informatik.uni-hamburg.de