COMP 551 – Applied Machine Learning
Lecture 5: Generative models for linear classification

Instructor: Joelle Pineau (*jpineau@cs.mcgill.ca)*

Class web page: *www.cs.mcgill.ca/~jpineau/comp551*

Today's quiz

- **Q1**. What is a **linear** classifier? (In contrast to a non-linear classifier.)

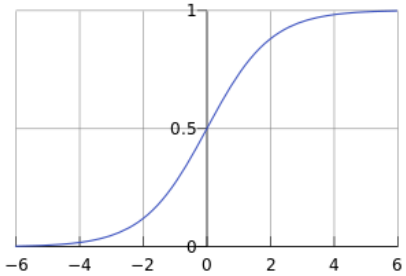
- **Q2.** Describe the difference between discriminative and generative classifiers.

- **Q3**. Consider the following data set. If you use logistic regression to compute a decision boundary, what is the prediction for x_6?

Data	Feature 1	Feature 2	Feature 3	Output
x_1	1	0	0	0
x_2	1	0	1	0
x_3	0	1	0	0
x_4	1	1	1	1
x_5	1	1	0	1
x_6	0	0	0	?

Quick recap

- Two approaches for linear classification:
  - **Discriminative learning**: Directly estimate $P(y|x)$.
    - **Logistic regression**, $P(y|x) : \sigma(WX) = 1 / (1 + e^{-WX})$



  - **Generative learning**: Separately model $P(x|y)$ and $P(y)$. Use these, through Bayes rule, to estimate $P(y|x)$.

Linear discriminant analysis (LDA)

- Return to Bayes rule: $P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)}$
- LDA makes explicit assumptions about *P(x|y)*: $P(x \mid y) = \frac{e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}}{(2\pi)^{1/2} \mid \Sigma \mid^{1/2}}$
  - Multivariate Gaussian, with mean *μ* and covariance matrix *Σ*.
  - Notation: here *x* is a single instance, represented as an *m*1* vector.
  - Key assumption of LDA: Both classes have the *same* covariance matrix, *Σ*.

Linear discriminant analysis (LDA)

- Return to Bayes rule: $P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)}$

- LDA makes explicit assumptions about *P(x|y)*: $P(x \mid y) = \frac{e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}}{(2\pi)^{1/2} \mid \Sigma \mid^{1/2}}$

  - Multivariate Gaussian, with mean μ and covariance matrix Σ.
  - Notation: here x is a single instance, represented as an $m*1$ vector.
  - Key assumption of LDA: Both classes have the *same* covariance matrix, Σ.

- Consider the log-odds ratio (again, *P(x)* doesn't matter for decision):

$$\ln \frac{P(x \mid y=1)P(y=1)}{P(x \mid y=0)P(y=0)} = \underbrace{\ln \frac{P(y=1)}{P(y=0)} - \frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1}_{w_0} + \underbrace{x^T \Sigma^{-1}(\mu_1 - \mu_2)}_{x^T w}$$

This is a linear decision boundary!

Learning in LDA: 2 class case

- Estimate $P(y)$, μ, Σ, from the training data, then apply log-odds ratio.

Learning in LDA: 2 class case

- Estimate $P(y)$, μ, Σ, from the training data, then apply log-odds ratio.
  - $P(y=0) = N_0 / (N_0 + N_1)$ $\quad\quad$ $P(y=1) = N_1 / (N_0 + N_1)$

    where N_1, N_0, be # of training samples from classes 1 and 0, respectively.

Learning in LDA: 2 class case

- Estimate $P(y)$, μ, Σ, from the training data, then apply log-odds ratio.
  - $P(y=0) = N_0 / (N_0 + N_1)$ $\quad\quad$ $P(y=1) = N_1 / (N_0 + N_1)$

    where N_1, N_0, be # of training samples from classes 1 and 0, respectively.
  - $\mu_0 = \sum_{i=1:n} I(y_i=0)\, x_i / N_0$ $\quad\quad$ $\mu_1 = \sum_{i=1:n} I(y_i=1)\, x_i / N_1$

    where $I(x)$ is the indicator function: $I(x)=0$ if $x=0$, $I(x)=1$ if $x=1$.

Learning in LDA: 2 class case

- Estimate $P(y)$, μ, Σ, from the training data, then apply log-odds ratio.
  - $P(y=0) = N_0 / (N_0 + N_1)$ $\quad\quad$ $P(y=1) = N_1 / (N_0 + N_1)$

    where N_1, N_0, be # of training samples from classes 1 and 0, respectively.
  - $\mu_0 = \sum_{i=1:n} I(y_i=0)\, x_i / N_0$ $\quad\quad$ $\mu_1 = \sum_{i=1:n} I(y_i=1)\, x_i / N_1$

    where $I(x)$ is the indicator function: $I(x)=0$ if $x=0$, $I(x)=1$ if $x=1$.
  - $\Sigma = \sum_{k=0:1} \sum_{i=1:n} I(y_i=0)\, (x_i - \mu_k)(x_i - \mu_k)^T / (N_0+N_1-N_k)$

Learning in LDA: 2 class case

- Estimate $P(y)$, μ, Σ, from the training data, then apply log-odds ratio.
  - $P(y=0) = N_0 / (N_0 + N_1)$ $\quad\quad$ $P(y=1) = N_1 / (N_0 + N_1)$

    where N_1, N_0, be # of training samples from classes 1 and 0, respectively.
  - $\mu_0 = \sum_{i=1:n} I(y_i=0)\, x_i / N_0$ $\quad\quad$ $\mu_1 = \sum_{i=1:n} I(y_i=1)\, x_i / N_1$

    where $I(x)$ is the indicator function: $I(x)=0$ if $x=0$, $I(x)=1$ if $x=1$.
  - $\Sigma = \sum_{k=0:1} \sum_{i=1:n} I(y_i=0)\, (x_i - \mu_k)(x_i - \mu_k)^T / (N_0+N_1-N_k)$

- **Decision boundary**: Given an input x, classify it as class 1 if the log-odds ratio is >0, classify it as class 0 otherwise.

More complex decision boundaries

- Want to accommodate more than 2 classes?
  - Consider the per-class linear discriminant function:

$$\delta_y(x) = x^T \Sigma^{-1} \mu_y - \frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y + \log P(y)$$

  - Use the following decision rule:

$$\text{Output} = \operatorname*{argmax}_y \delta_y(x) = \operatorname*{argmax}_y \left[x^T \Sigma^{-1} \mu_y - \frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y + \log P(y) \right]$$

- Want more flexible (non-linear) decision boundaries?
  - Recall trick from linear regression of re-coding variables.

Using higher-order features

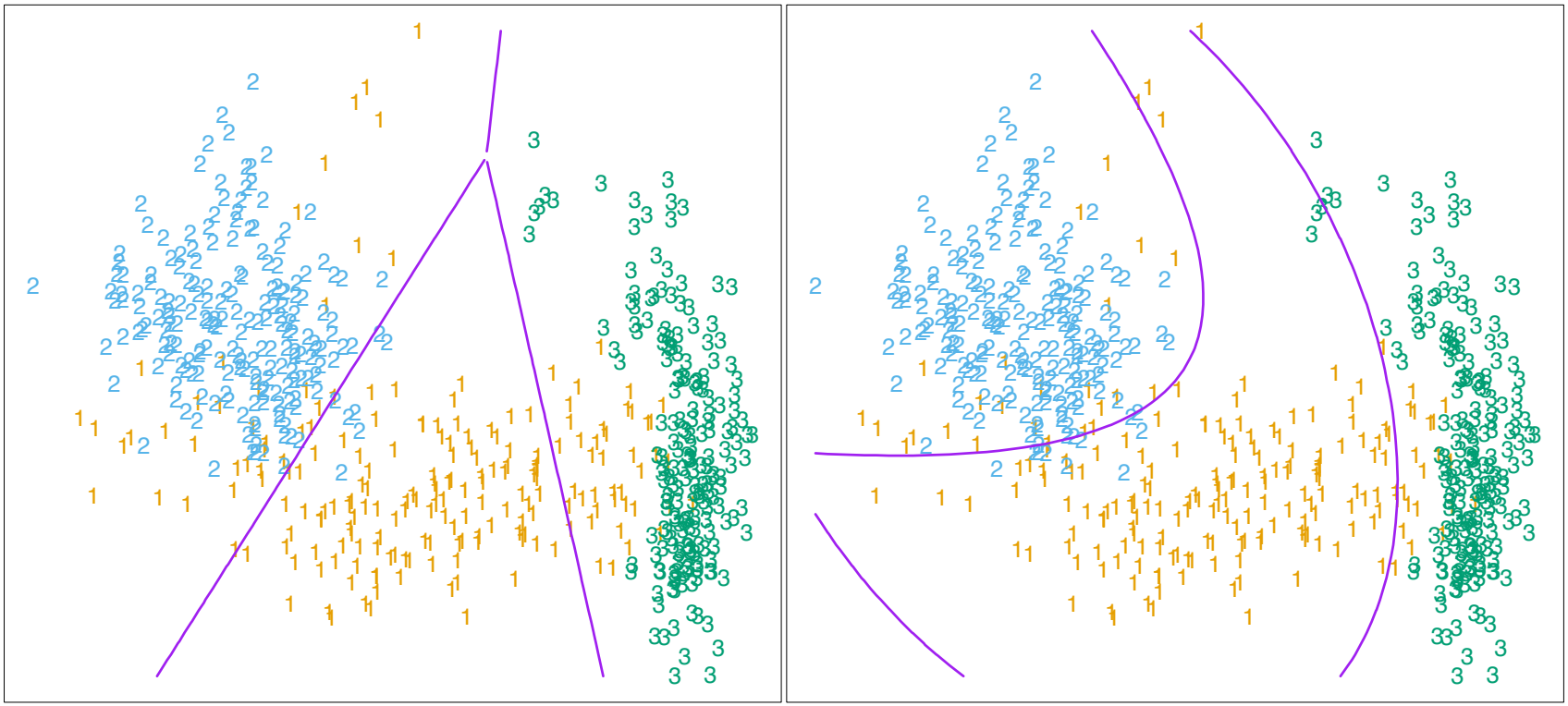


FIGURE 4.1. *The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space* $X_1, X_2, X_1X_2, X_1^2, X_2^2$. *Linear inequalities in this space are quadratic inequalities in the original space.*

Quadratic discriminant analysis

- LDA assumes all classes have the same covariance matrix, Σ.
- QDA allows different covariance matrices, Σ_y for each class y.

- Linear discriminant function:

$$\delta_y(x) = x^T \Sigma^{-1} \mu_y - \frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y + \log P(y)$$

- Quadratic discriminant function:

$$\delta_y(x) = -\frac{1}{2} \left| \Sigma_y \right| - \frac{1}{2} (x - \mu_y)^T \Sigma_y^{-1} (x - \mu_y) + \log P(y)$$

  - QDA has more parameters to estimate, but greater flexibility to estimate the target function. Bias-variance trade-off.

LDA vs QDA

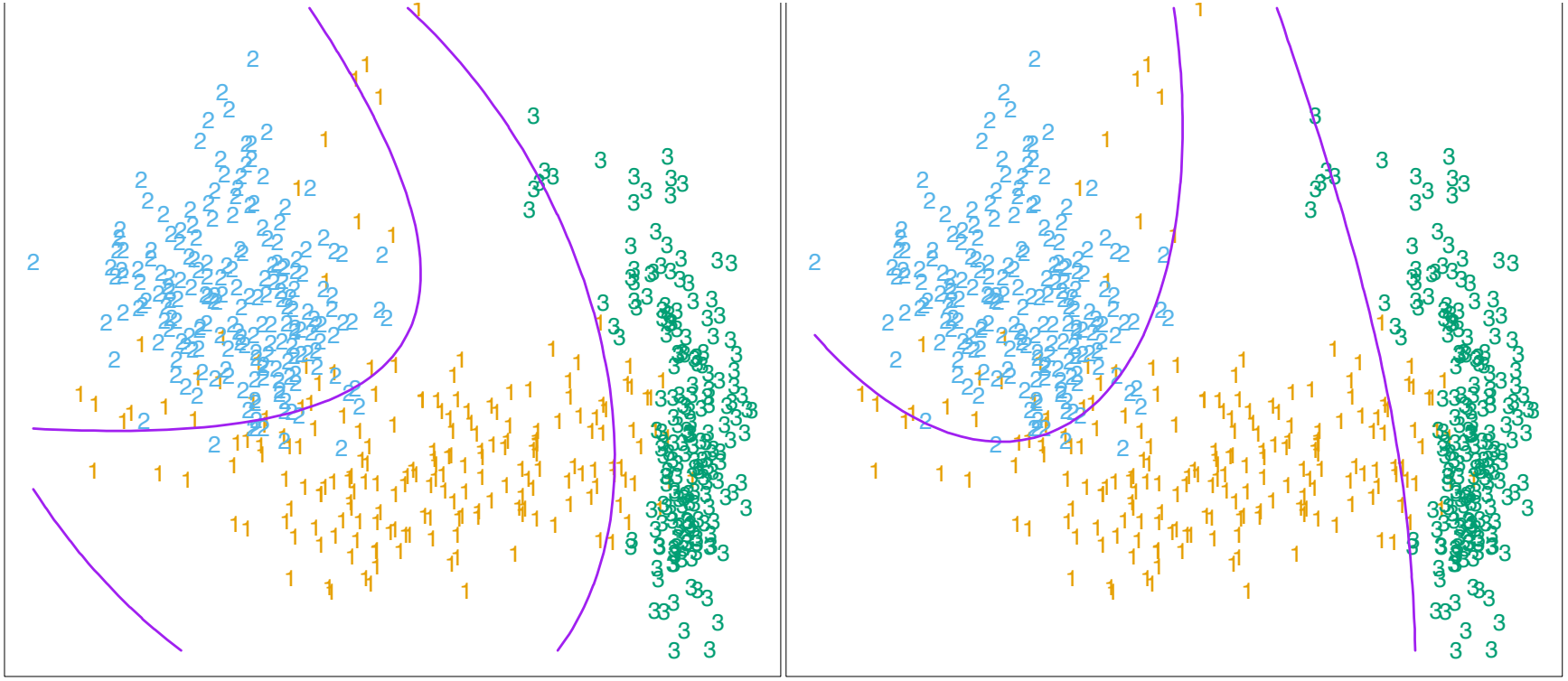


FIGURE 4.6. *Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $X_1, X_2, X_1X_2, X_1^2, X_2^2$). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is*

Generative learning – Scaling up

- Consider applying generative learning (e.g. LDA) to Project 1 data.

Generative learning – Scaling up

- Consider applying generative learning (e.g. LDA) to Project 1 data.

  E.g. Task: Given first few words, determine language (without dictionary).

  – What are features? How to encode the data?

  – How to handle high dimensional feature space?

  – Assume same co-variance matrix for both classes? (Maybe that is ok.)

What kinds of assumptions can we make to simplify the problem?

Naïve Bayes assumption

- **Generative learning**: Estimate $P(x|y)$, $P(y)$. Then calculate $P(y|x)$.

- **Naïve Bayes**: Assume the x_j are conditionally independent given y.
  - In other words: $P(x_j \mid y) = P(x_j \mid y, x_k)$, for all j, k

Naïve Bayes assumption

- **Generative learning**: Estimate $P(x|y)$, $P(y)$. Then calculate $P(y|x)$.

- **Naïve Bayes**: Assume the x_j are conditionally independent given y.
  - In other words: $P(x_j \mid y) = P(x_j \mid y, x_k)$, for all j, k

- Generative model structure:

$$P(x \mid y) = P(x_1, x_2, \ldots, x_m \mid y)$$

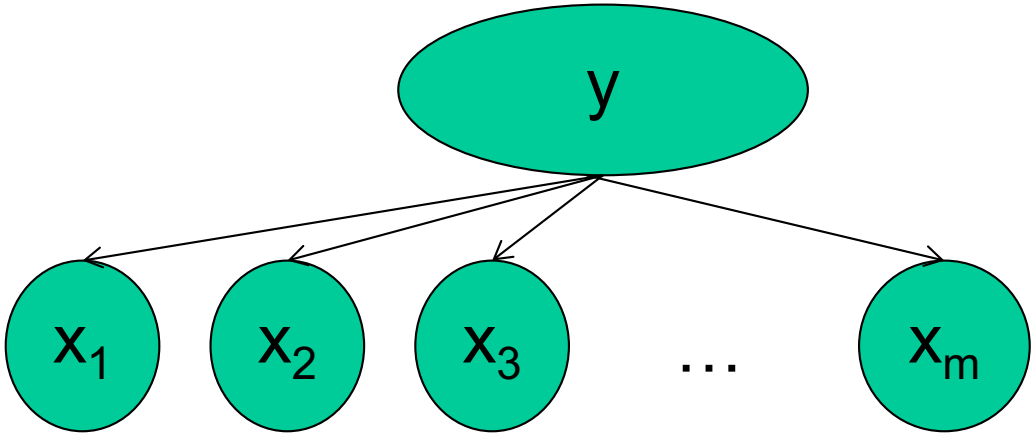
$$= P(x_1 \mid y)\, P(x_2 \mid y, x_1)\, P(x_3 \mid y, x_1, x_2) \ldots P(x_m \mid y, x_1, x_2, \ldots, x_{m-1})$$

(from general rules of probabilities)

$$= P(x_1 \mid y)\, P(x_2 \mid y)\, P(x_3 \mid y) \ldots P(x_m \mid y)$$

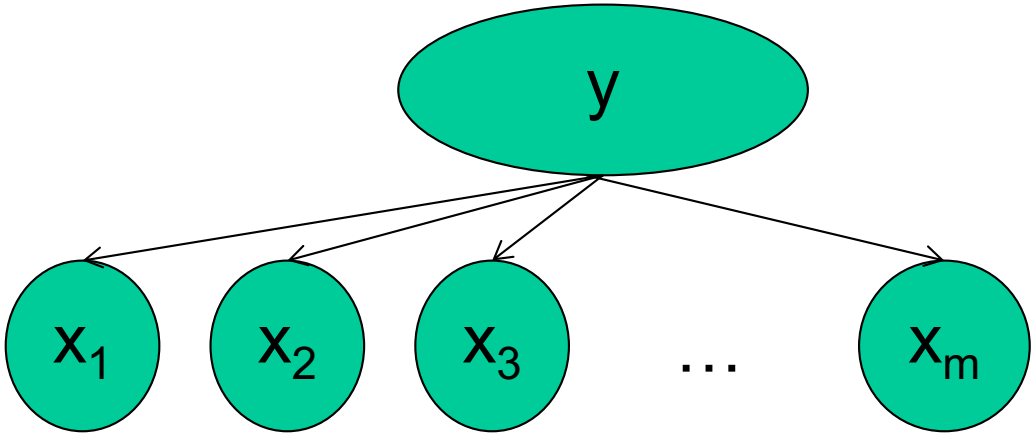
(from the Naïve Bayes assumption above)

Naïve Bayes graphical model



- How many parameters to estimate? Assume *m* binary features.

Naïve Bayes graphical model



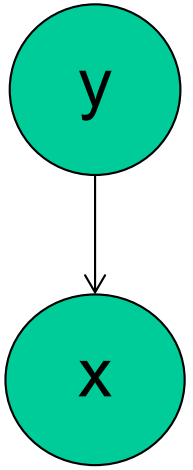
- How many parameters to estimate? Assume *m* binary features.
  - without Naïve Bayes assumption: $O(2^m)$ numbers to describe model.
  - With Naïve Bayes assumption: $O(m)$ numbers to describe model.
- **Useful when the number of features is high.**

Training a Naïve Bayes classifier

- Assume *x, y* are binary variables, *m=1*.
- Estimate the parameters *P(x|y)* and *P(y)* from data.
  - Define: $\theta_1 = Pr\ (y=1)$

    $\theta_{j,1} = Pr\ (x_j=1 \mid y=1)$

    $\theta_{j,0} = Pr\ (x_j=1 \mid y=0)$.

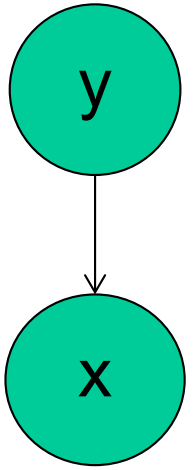


Training a Naïve Bayes classifier

- Assume *x, y* are binary variables, *m=1*.
- Estimate the parameters *P(x|y)* and *P(y)* from data.
  - Define: $\theta_1 = Pr\ (y=1)$

    $\theta_{j,1} = Pr\ (x_j=1 \mid y=1)$

    $\theta_{j,0} = Pr\ (x_j=1 \mid y=0)$.



- **Evaluation criteria**: Find parameters that maximize the log-likelihood function.

- Likelihood: $Pr(y|x) \propto Pr(y)Pr(x|y) = \prod_{i=1:n} (P(y_i) \prod_{j=1:m} P(x_{i,j} \mid y_i))$
  - Samples *i* are independent, so we take product over *n*.
  - Input features are independent (cond. on *y*) so we take product over *m*.

Training a Naïve Bayes classifier

- Likelihood for binary output variable: $L(\theta_1|y=1) = \theta_1^y(1-\theta_1)^{1-y}$
- Log-likelihood for all parameters:

$$\log L(\theta_1, \theta_{i,1}, \theta_{i,0} \mid D) = \Sigma_{i=1:n} [\log P(y_i) + \Sigma_{j=1:m} \log P(x_{i,j} \mid y_i)]$$

Training a Naïve Bayes classifier

- Likelihood for binary output variable: $L(\theta_1|y=1) = \theta_1^{y}(1-\theta_1)^{1-y}$
- Log-likelihood for all parameters:

$$\log L(\theta_1, \theta_{i,1}, \theta_{i,0} \mid D) = \Sigma_{i=1:n} [\log P(y_i) + \Sigma_{j=1:m} \log P(x_{\mathbf{i,j}} \mid y_i)]$$

$$= \Sigma_{i=1:n} [\, y_i \log \theta_1 + (1-y_i)\log(1-\theta_1)$$

$$+ \Sigma_{j=1:m} y_i \, (x_{i,j} \log\theta_{i,1} + (1-x_{i,j})\log(1-\theta_{i,1}))$$

$$+ \Sigma_{j=1:m} (1-y_i)(x_{i,j} \log\theta_{i,0} + (1-x_{i,j})\log(1-\theta_{i,0}))]$$

Training a Naïve Bayes classifier

- Likelihood for binary output variable: $L(\theta_1|y=1) = \theta_1^{y}(1-\theta_1)^{1-y}$
- Log-likelihood for all parameters:

$$\log L(\theta_1, \theta_{i,1}, \theta_{i,0} \mid D) = \Sigma_{i=1:n} [\log P(y_i) + \Sigma_{j=1:m} \log P(x_{\mathbf{i,j}} \mid y_i)]$$

$$= \Sigma_{i=1:n} [\, y_i \log \theta_1 + (1-y_i)\log(1-\theta_1)$$

$$+ \Sigma_{j=1:m} y_i \, (x_{i,j} \log\theta_{i,1} + (1-x_{i,j})\log(1-\theta_{i,1}))$$

$$+ \Sigma_{j=1:m} (1-y_i)(x_{i,j} \log\theta_{i,0} + (1-x_{i,j})\log(1-\theta_{i,0}))]$$

(will have other form if params $P(x|y)$ have other form, e.g. Gaussian).

Training a Naïve Bayes classifier

- Likelihood for binary output variable: $L(\theta_1|y=1) = \theta_1^y(1-\theta_1)^{1-y}$
- Log-likelihood for all parameters:

$$\log L(\theta_1, \theta_{i,1}, \theta_{i,0} \mid D) = \Sigma_{i=1:n} [\log P(y_i) + \Sigma_{j=1:m} \log P(x_{i,j} \mid y_i)]$$

$$= \Sigma_{i=1:n} [y_i \log \theta_1 + (1-y_i)\log(1-\theta_1)$$

$$+ \Sigma_{j=1:m} y_i (x_{i,j} \log\theta_{i,1} + (1-x_{i,j})\log(1-\theta_{i,1}))$$

$$+ \Sigma_{j=1:m} (1-y_i)(x_{i,j} \log\theta_{i,0} + (1-x_{i,j})\log(1-\theta_{i,0}))]$$

(will have other form if params $P(x|y)$ have other form, e.g. Gaussian).

- To estimate θ_1, take derivative of $\log L$ with respect to θ_1, set to 0:

$$\partial L / \partial \theta_1 = \Sigma_{i=1:n} (y_i/\theta_1 - (1-y_i)/(1-\theta_1)) = 0$$

Training a Naïve Bayes classifier

Solving for θ_1 we get:

$\theta_1 = (1/n) \Sigma_{i=1:n} y_i$

= number of examples where $y=1$ / number of examples

Training a Naïve Bayes classifier

Solving for θ_1 we get:

$\theta_1 = (1/n) \Sigma_{i=1:n} y_i$

= number of examples where $y=1$ / number of examples

Similarly, we get:

$\theta_{j,1}$ = number of examples where $x_j=1$ and $y=1$ / number of examples where $y=1$

$\theta_{j,0}$ = number of examples where $x_j=1$ and $y=0$ / number of examples where $y=0$

Naïve Bayes decision boundary

- Consider again the log-odds ratio:

$$\log \frac{\Pr(y=1 \mid x)}{\Pr(y=0 \mid x)} = \log \frac{\Pr(x \mid y=1)P(y=1)}{\Pr(x \mid y=0)P(y=0)}$$

$$= \log \frac{P(y=1)}{P(y=0)} + \log \frac{\prod_{j=1}^{m} P\left(x_j \mid y=1\right)}{\prod_{j=1}^{m} P\left(x_j \mid y=0\right)}$$

$$= \log \frac{P(y=1)}{P(y=0)} + \sum_{j=1}^{m} \log \frac{P\left(x_j \mid y=1\right)}{P\left(x_j \mid y=0\right)}$$

Naïve Bayes decision boundary

- Consider the case where features are binary: $x_j = \{0, 1\}$
- Define:

$$w_{j,0} = \log \frac{P(x_j = 0 \mid y = 1)}{P(x_j = 0 \mid y = 0)}; \qquad w_{j,1} = \log \frac{P(x_j = 1 \mid y = 1)}{P(x_j = 1 \mid y = 0)}$$

- Now we have:

$$\log \frac{\Pr(y = 1 \mid x)}{\Pr(y = 0 \mid x)} = \log \frac{P(y = 1)}{P(y = 0)} + \sum_{j=1}^{m} \log \frac{P\left(x_j \mid y = 1\right)}{P\left(x_j \mid y = 0\right)}$$

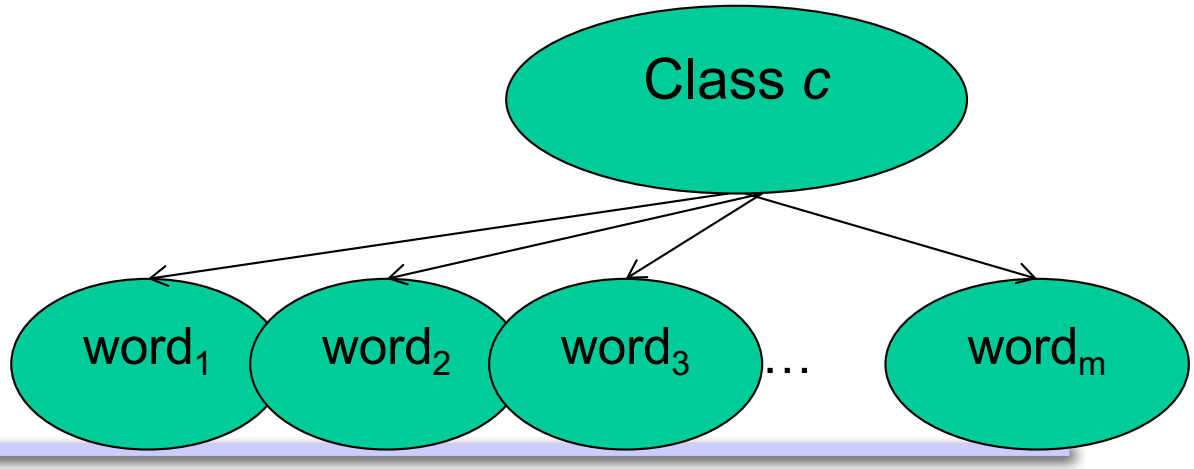
$$= \log \frac{P(y = 1)}{P(y = 0)} + \sum_{j=1}^{m} (w_{j,0}(1 - x_j) + w_{j,1} x_j)$$

$$= \underbrace{\log \frac{P(y = 1)}{P(y = 0)} + \sum_{j=1}^{m} w_{j,0}}_{w_0} + \underbrace{\sum_{j=1}^{m} (w_{j,1} - w_{j,0}) x_j}_{x^T w}$$

This is a linear decision boundary!

Text classification example

- Using Naïve Bayes, we can compute probabilities for all the words which appear in the document collection.
  - $P(y=c)$ is the probability of class c
  - $P(x_j \mid y=c)$ is the probability of seeing word j in documents of class c

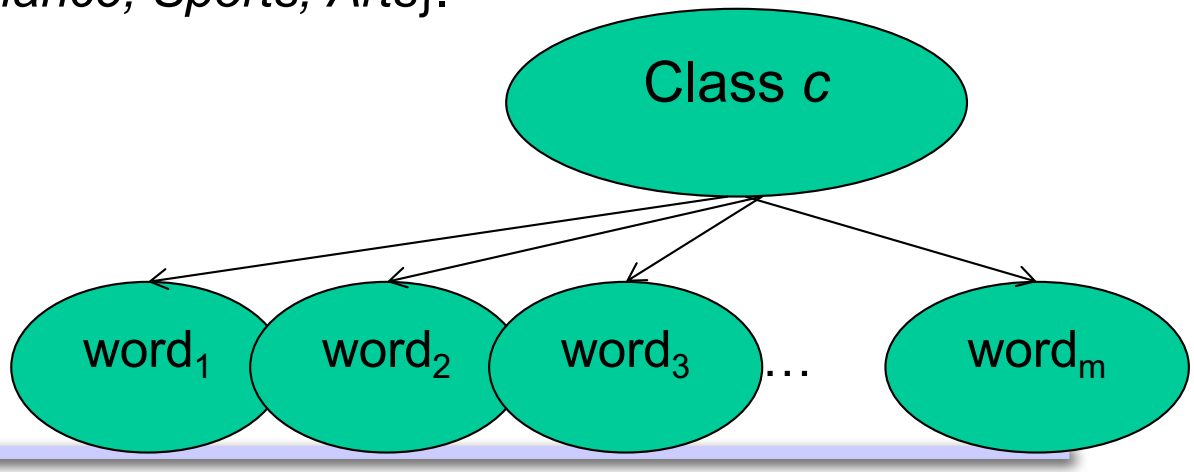


Text classification example

- Using Naïve Bayes, we can compute probabilities for all the words which appear in the document collection.
  - $P(y=c)$ is the probability of class c
  - $P(x_j \mid y=c)$ is the probability of seeing word j in documents of class c

- Set of classes depends on the application, e.g.
  - Topic modeling: each class corresponds to documents on a given topic, e.g. {*Politics, Finance, Sports, Arts*}.

What happens when a word is not observed in the training data?



Laplace smoothing

- Replace the maximum likelihood estimator:

$$Pr(x_j \mid y=1) = \frac{\text{number of instance with } x_j=1 \text{ and } y=1}{\text{number of examples with } y=1}$$

Laplace smoothing

- Replace the maximum likelihood estimator:

$$Pr(x_j \mid y=1) = \frac{\text{number of instance with } x_j=1 \text{ and } y=1}{\text{number of examples with } y=1}$$

- With the following:

$$Pr(x_j \mid y=1) = \frac{(\text{number of instance with } x_j=1 \text{ and } y=1) + 1}{(\text{number of examples with } y=1) + 2}$$

Laplace smoothing

- Replace the maximum likelihood estimator:

$$Pr(x_j \mid y=1) = \frac{\text{number of instance with } x_j=1 \text{ and } y=1}{\text{number of examples with } y=1}$$

- With the following:

$$Pr(x_j \mid y=1) = \frac{(\text{number of instance with } x_j=1 \text{ and } y=1) + 1}{(\text{number of examples with } y=1) + 2}$$

  - If no example from that class, it reduces to a prior probability or $Pr=1/2$.
  - If all examples have $x_j=1$, then $Pr(x_j=0|y)$ has $Pr = 1 / (\#examples + 1)$.
  - If a word appears frequently, the new estimate is only slightly biased.

- This is an example of **Bayesian prior** for Naïve Bayes.

Example: 20 newsgroups

- Given 1000 training documents from each group, learn to classify new documents according to which newsgroup they came from:

comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.baseball
comp.windows.x	rec.sport.hockey
alt.atheism	sci.space
soc.religion.christian	sci.crypt
talk.religion.misc	sci.electronics
talk.politics.mideast	sci.med
talk.politics.misc	talk.politics.guns

- Naïve Bayes: 89% classification accuracy (comparable to other state-of-the-art methods.)

Multi-class classification

- Generally two options:
  1. Learn a single classifier that can produce 20 distinct output values.
  2. Learn 20 different 1-vs-all binary classifiers.

Multi-class classification

- Generally two options:
  1. Learn a single classifier that can produce 20 distinct output values.
  2. Learn 20 different 1-vs-all binary classifiers.

- Option 1 assumes you have a multi-class version of the classifier.
  - For Naïve Bayes, compute $P(y|x)$ for each class, and select the class with highest probability.

Multi-class classification

- Generally two options:
  1. Learn a single classifier that can produce 20 distinct output values.
  2. Learn 20 different 1-vs-all binary classifiers.

- Option 1 assumes you have a multi-class version of the classifier.
  - For Naïve Bayes, compute $P(y|x)$ for each class, and select the class with highest probability.

- Option 2 applies to all binary classifiers, so more flexible. But often slower (need to learn many classifiers), and creates a class imbalance problem (the target class has relatively fewer data points, compared to the aggregation of the other classes.)

Best features extracted

category	word	log-odds ratio	$\hat{\theta}_k$
alt.atheism	atheism	0.013	0.0040
comp.graphics	jpeg	0.037	0.0073
comp.os.ms-windows.misc	windows	0.043	0.020
comp.sys.ibm.pc.hardware	scsi	0.033	0.012
comp.sys.mac.hardware	mac	0.024	0.012
comp.windows.x	window	0.024	0.0091
misc.forsale	sale	0.018	0.0076
rec.autos	car	0.043	0.017
rec.motorcycles	bike	0.045	0.010
rec.sport.baseball	baseball	0.016	0.0057
rec.sport.hockey	hockey	0.037	0.0078
sci.crypt	clipper	0.033	0.0058
sci.electronics	circuit	0.010	0.0031
sci.med	patients	0.011	0.0029
sci.space	space	0.035	0.013
soc.religion.christian	god	0.035	0.018
talk.politics.guns	gun	0.028	0.0094
talk.politics.mideast	armenian	0.039	0.0057
talk.politics.misc	stephanopoulos	0.024	0.0034
talk.religion.misc	god	0.011	0.011

Table 3.1: For each category in the 20 Newsgroups dataset, the word with the highest log odds ratio. A larger score indicates a word which is commonly found in the specified category, but rarely found in other categories. Words with high log odds ratios are good discriminants for the one vs. all problem.

From MSc thesis by Jason Rennie *http://qwone.com/~jason/papers/sm-thesis.pdf*

Gaussian Naïve Bayes

Extending Naïve Bayes to continuous inputs:

- $P(y)$ is still assumed to be a binomial distribution.
- $P(x|y)$ is assumed to be a multivariate Gaussian (normal) distribution with mean $\mu \in \Re^n$ and covariance matrix $\Sigma \in \Re^n x \Re^n$

Gaussian Naïve Bayes

Extending Naïve Bayes to continuous inputs:

- $P(y)$ is still assumed to be a binomial distribution.
- $P(x|y)$ is assumed to be a multivariate Gaussian (normal) distribution with mean $\mu \in \Re^n$ and covariance matrix $\Sigma \in \Re^n x \Re^n$
  - If we assume the same Σ for all classes: Linear discriminant analysis.
  - If Σ is distinct between classes: Quadratic discriminant analysis.
  - If Σ is diagonal (i.e. features are independent): Gaussian Naïve Bayes.

Gaussian Naïve Bayes

Extending Naïve Bayes to continuous inputs:

- $P(y)$ is still assumed to be a binomial distribution.
- $P(x|y)$ is assumed to be a multivariate Gaussian (normal) distribution with mean $\mu \in \Re^n$ and covariance matrix $\Sigma \in \Re^n x \Re^n$
  - If we assume the same Σ for all classes: Linear discriminant analysis.
  - If Σ is distinct between classes: Quadratic discriminant analysis.
  - If Σ is diagonal (i.e. features are independent): Gaussian Naïve Bayes.

- How do we estimate parameters? Derive the maximum likelihood estimators for μ and Σ.

More generally: Bayesian networks

- If not all conditional independence relations are true, we can introduce new arcs to show what dependencies are there.
- At each node, we have the conditional probability distribution of the corresponding variable, given its parents.
- This more general type of graph, annotated with conditional distributions, is called a Bayesian network.
- More on this in COMP-652.

What you should know

- Naïve Bayes assumption
- Log-odds ratio decision boundary
- How to estimate parameters for Naïve Bayes
- Laplace smoothing
- Relation between Naïve Bayes, LDA, QDA, Gaussian Naïve Bayes.