DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

February 21 2019

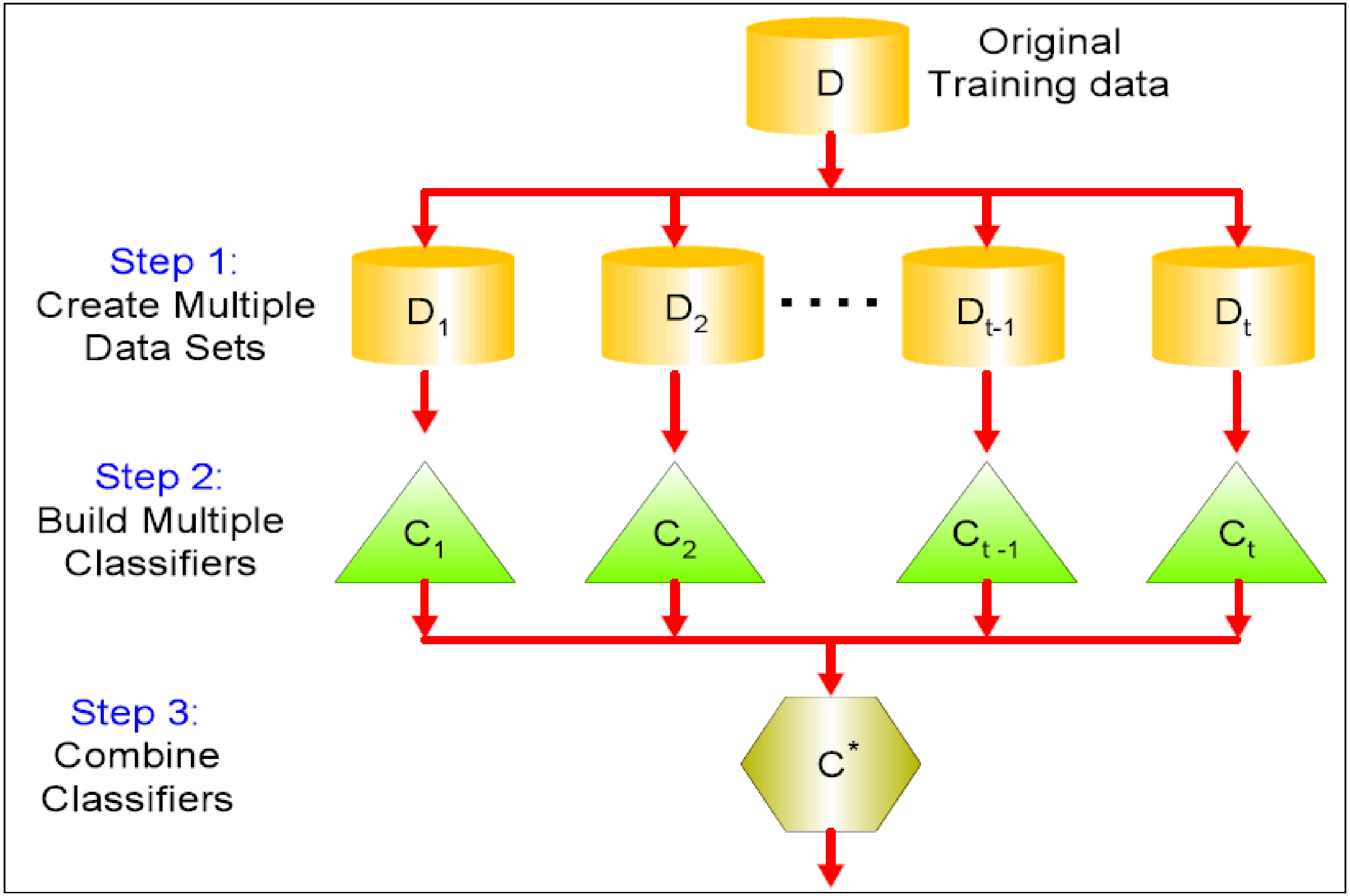
Logistics

- HW3 is due on Friday, February 22
- Project proposal due on Tuesday 02/26 on Gradescope
  - Project Title
  - Problem Description
  - Dataset
  - Feature extraction and selection
  - ML algorithms
  - Metrics for evaluation
- Week of February 25
  - Lecture on 02/26 taught by Lisa Friedland
  - Lecture on 02/28 canceled

Review

- Ensemble learning are powerful learning methods
  - Better accuracy than standard classifiers
- Bagging uses bootstrapping (with replacement), trains T models, and averages their prediction
  - Random forests vary training data and feature set at each split
- Boosting is an ensemble of T weak learners that emphasizes mis-predicted examples
  - AdaBoost has great theoretical and experimental performance
  - Can be used with linear models or simple decision trees (stumps, fixed-depth decision trees)

Bagging

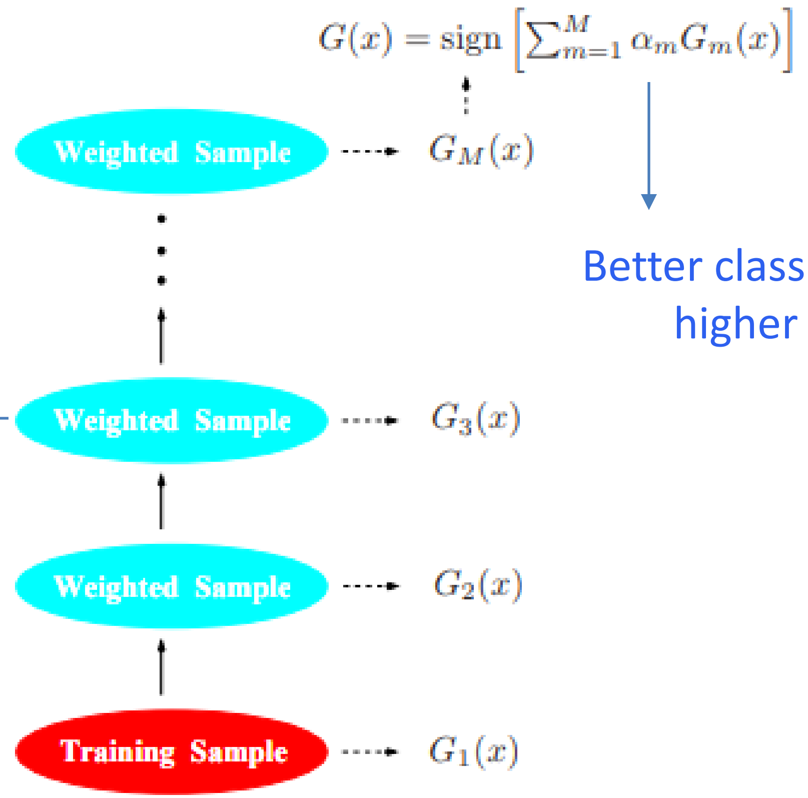


Majority Votes

Overview of AdaBoost

- Mis-classified examples get higher weights
- Correct examples get lower weights

Better classifiers will get higher weights



Uniform weights

FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Bagging vs Boosting

Bagging	**vs.**	**Boosting**
Resamples data points		Reweights data points (modifies their distribution)
Weight of each classifier is the same		Weight is dependent on classifier's accuracy
Only variance reduction		Both bias and variance reduced – learning rule becomes more complex with iterations

Outline

- Density Estimation
  - Estimating prior and joint probabilities
  - Risk of overfitting
- Naïve Bayes classifier
- Application
  - Document classification

Essential probability concepts

- Marginalization: $P(B) = \sum_{v \in \text{values}(A)} P(B \wedge A = v)$
- Conditional Probability: $P(A \mid B) = \dfrac{P(A \wedge B)}{P(B)}$
- Bayes' Rule: $P(A \mid B) = \dfrac{P(B \mid A) \times P(A)}{P(B)}$
- Independence:

$$
\begin{aligned}
A \perp\!\!\!\perp B \quad &\leftrightarrow \quad P(A \wedge B) = P(A) \times P(B) \\
&\leftrightarrow \quad P(A \mid B) = P(A) \\
A \perp\!\!\!\perp B \mid C \quad &\leftrightarrow \quad P(A \wedge B \mid C) = P(A \mid C) \times P(B \mid C)
\end{aligned}
$$

Prior and Joint Probabilities

- **Prior probability**: degree of belief without any other evidence
- **Joint probability**: matrix of combined probabilities of a set of variables

Russell & Norvig's Alarm Domain: (boolean RVs)

- A world has a specific instantiation of variables:

  $(\mathrm{alarm} \wedge \mathrm{theft} \wedge \neg \mathrm{earthquake})$

- The joint probability is given by:

P(Alarm, Theft) =

	alarm	¬alarm
theft	0.09	0.01
¬theft	0.1	0.8

Computing Prior Probabilities

	alarm		¬alarm	
	earthquake	¬earthquake	earthquake	¬earthquake
theft	0.01	0.08	0.001	0.009
¬theft	0.01	0.09	0.01	0.79

$$P(alarm) = \sum_{b,e} P(alarm \wedge \text{theft} = b \wedge \mathrm{Earthquake} = e)$$

$$= 0.01 + 0.08 + 0.01 + 0.09 = 0.19$$

$$P(\text{theft}) = \sum_{a,e} P(\mathrm{Alarm} = a \wedge \text{theft} \wedge \mathrm{Earthquake} = e)$$

$$= 0.01 + 0.08 + 0.001 + 0.009 = 0.1$$

The Joint Distribution

Recipe for making a joint distribution of d variables:

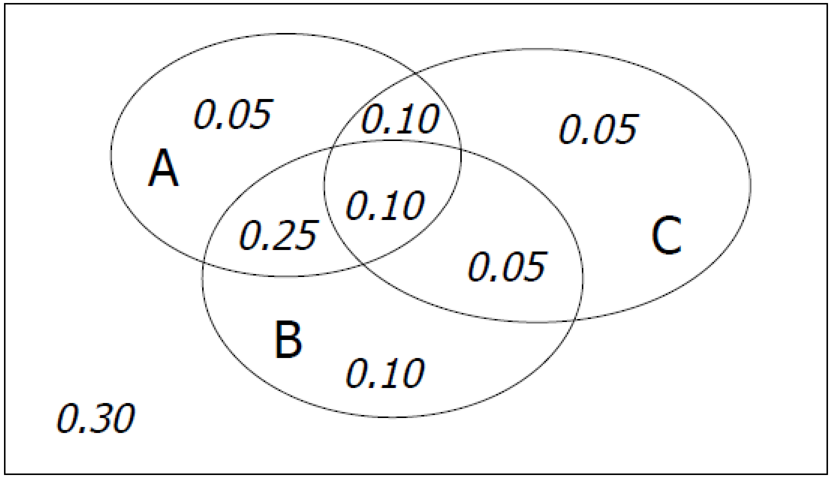
1. Make a truth table listing all combinations of values of your variables (if there are d Boolean variables then the table will have 2^d rows).

2. For each combination of values, say how probable it is.

3. If you subscribe to the axioms of probability, those numbers must sum to 1.

e.g., Boolean variables A, B, C

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



Learning Joint Distributions

Step 1:

Build a JD table for your attributes in which the probabilities are unspecified

A	B	C	Prob
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Step 2:

Then, fill in each row with:

$$\hat{P}(\text{row}) = \frac{\text{records matching row}}{\text{total number of records}}$$

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	**0.25**
1	1	1	0.10

Fraction of all records in which A and B are true but C is false

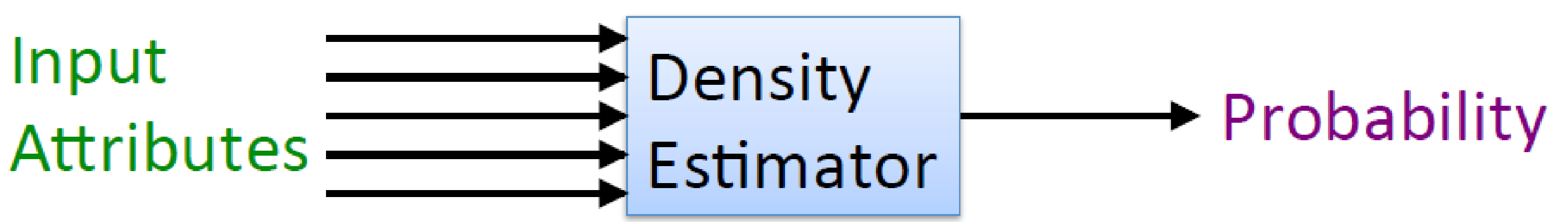
Example – Learning Joint Probability Distribution

This Joint PD was obtained by learning from three attributes in the UCI "Adult" Census Database [Kohavi 1995]

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

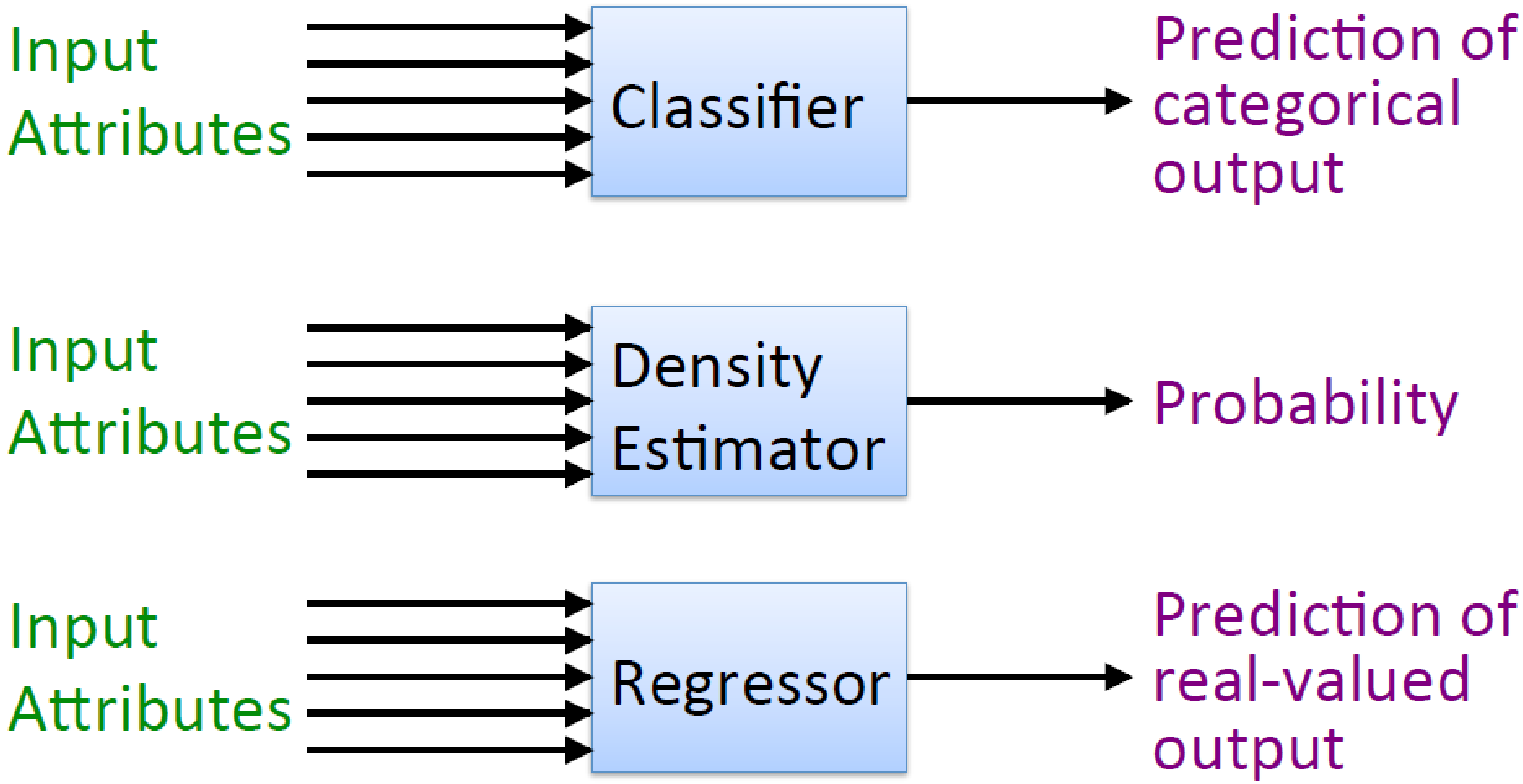
Density Estimation

- Our joint distribution learner is an example of something called **Density Estimation**
- A Density Estimator learns a mapping from a set of attributes to a probability

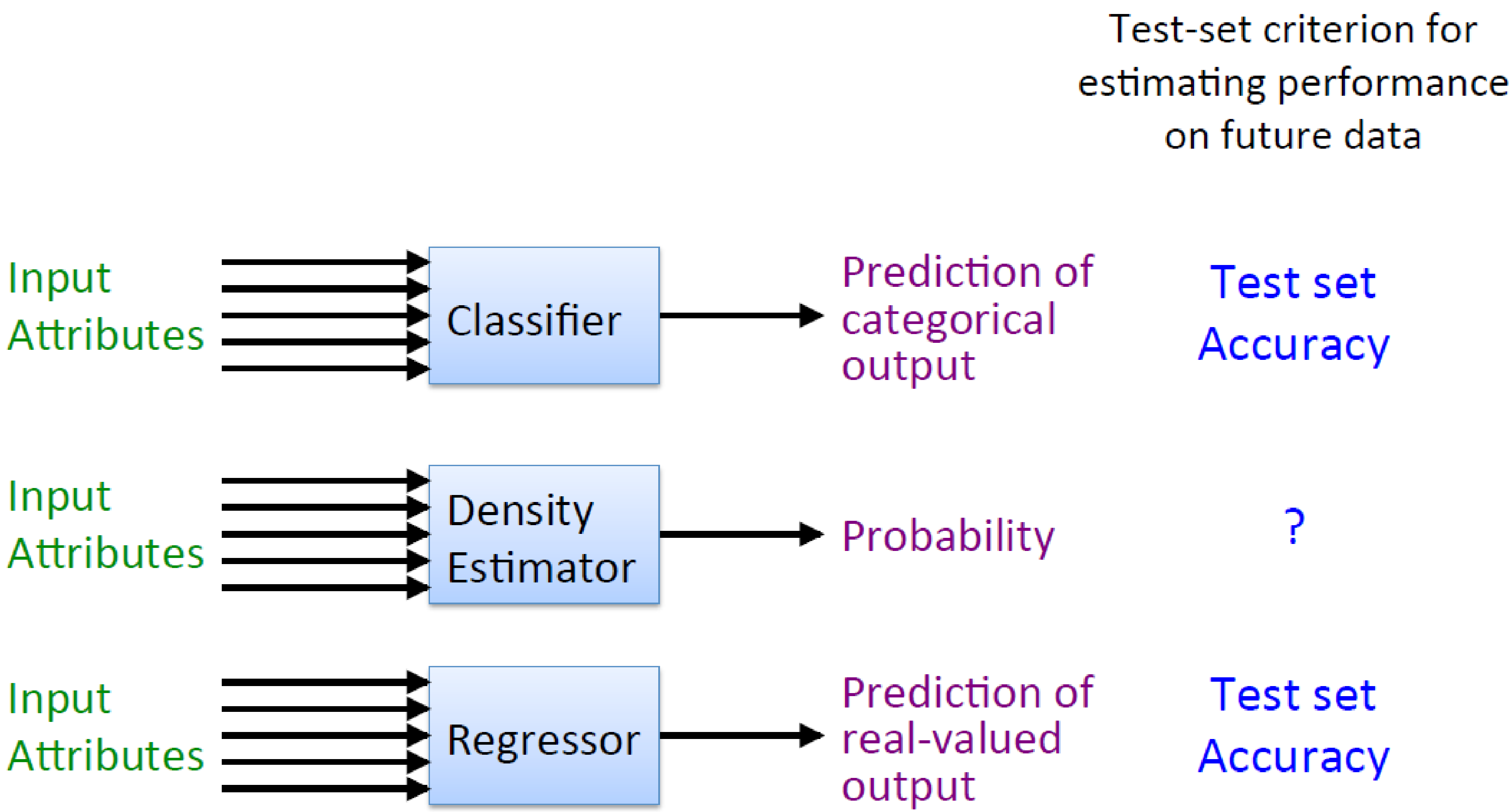


Density Estimation

Compare it against the two other major kinds of models:



Evaluating Density Estimators



Evaluating Density Estimators

- Given a record $\mathbf{x}$, a density estimator *M* can tell you how likely the record is:

$$\hat{P}(\mathbf{x} \mid M)$$

- The density estimator can also tell you how likely the dataset is:
  - Under the assumption that all records were independently generated from the Density Estimator's JD (that is, i.i.d.)

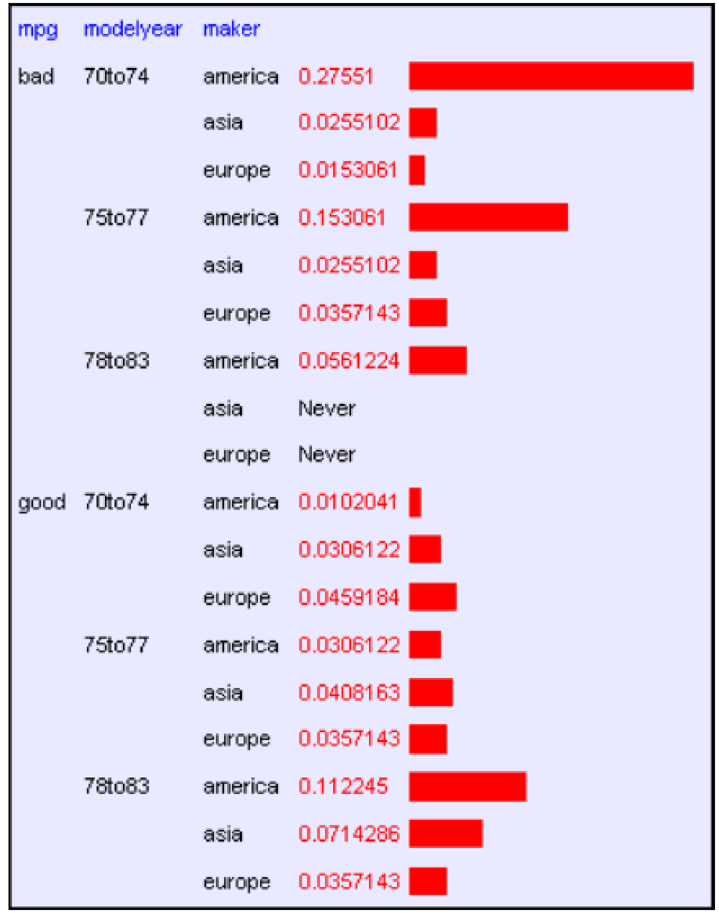
$$\hat{P}(\underbrace{\mathbf{x}_1 \wedge \mathbf{x}_2 \wedge \ldots \wedge \mathbf{x}_n}_{\text{dataset}} \mid M) = \prod_{i=1}^{n} \hat{P}(\mathbf{x}_i \mid M)$$

Example

From the UCI repository (thanks to Ross Quinlan)

- 192 records in the training set

mpg	modelyear	maker
good	75to78	asia
bad	70to74	america
bad	75to78	europe
bad	70to74	america
bad	70to74	america
bad	70to74	asia
bad	70to74	asia
bad	75to78	america
:	:	:
:	:	:
:	:	:
bad	70to74	america
good	79to83	america
bad	75to78	america
good	79to83	america
bad	75to78	america
good	79to83	america
good	79to83	america
bad	70to74	america
good	75to78	europe
bad	75to78	europe



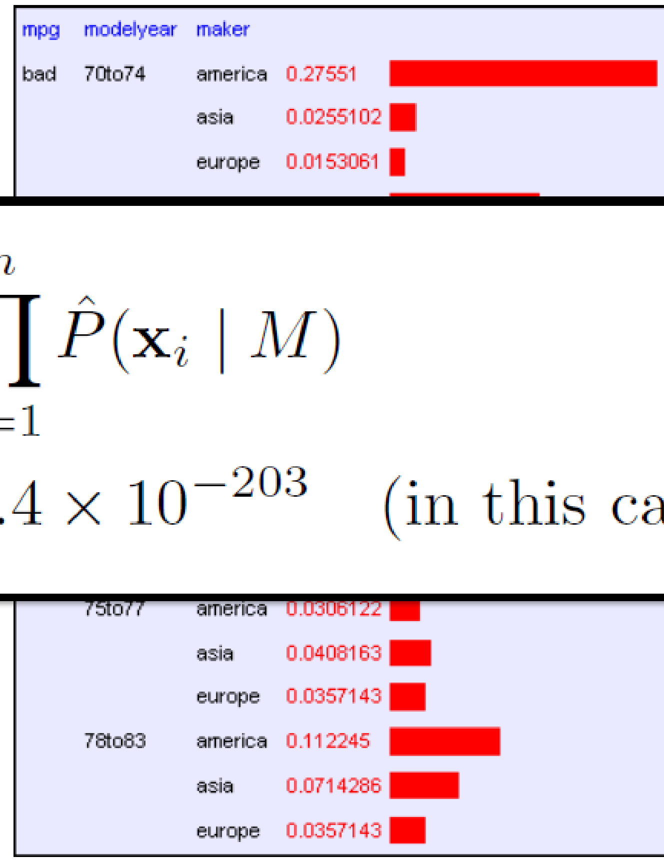
e by Andrew Moore

Example

From the UCI repository (thanks to Ross Quinlan)

- 192 records in the training set

mpg	modelyear	maker
good	75to78	asia
bad	70to74	america
bad	75to78	america
good	79to83	america
good	79to83	america
bad	70to74	america
good	75to78	europe
bad	75to78	europe



$$\hat{P}(\text{dataset} \mid M) = \prod_{i=1}^{n} \hat{P}(\mathbf{x}_i \mid M)$$

$$= 3.4 \times 10^{-203} \quad \text{(in this case)}$$

Slide by Andrew Moore

Log Probabilities

- For decent sized data sets, this product will underflow

$$\hat{P}(\text{dataset} \mid M) = \boxed{\prod_{i=1}^{n}} \hat{P}(\mathbf{x}_i \mid M)$$

- Therefore, since probabilities of datasets get so small, we usually use log probabilities

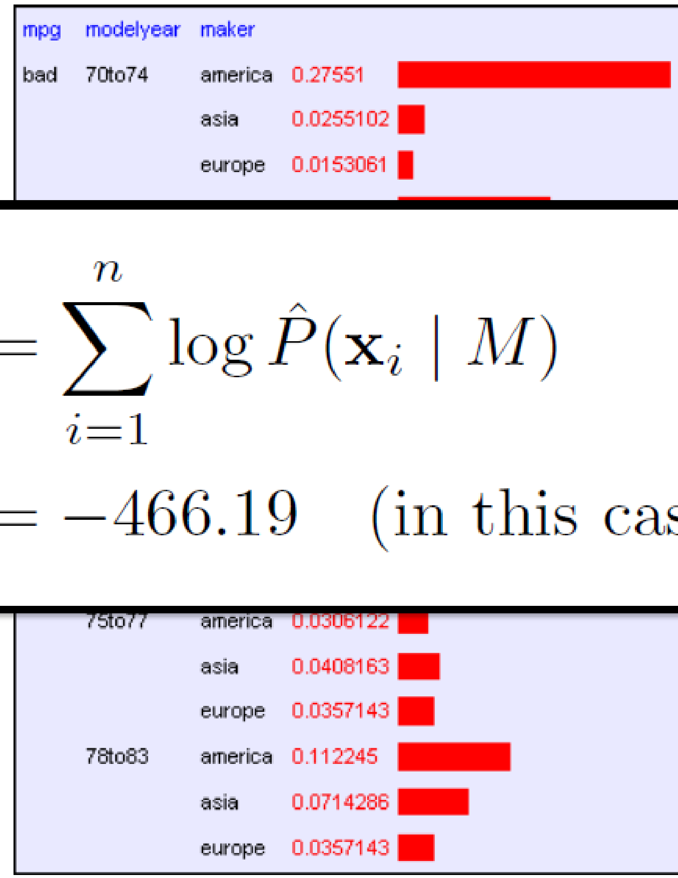
$$\log \hat{P}(\text{dataset} \mid M) = \log \prod_{i=1}^{n} \hat{P}(\mathbf{x}_i \mid M) = \sum_{i=1}^{n} \log \hat{P}(\mathbf{x}_i \mid M)$$

Example

From the UCI repository (thanks to Ross Quinlan)

- 192 records in the training set

mpg	modelyear	maker
good	75to78	asia
bad	70to74	america
bad	75to78	america
good	79to83	america
good	79to83	america
bad	70to74	america
good	75to78	europe
bad	75to78	europe



$$\log \hat{P}(\text{dataset} \mid M) = \sum_{i=1}^{n} \log \hat{P}(\mathbf{x}_i \mid M)$$
$$= -466.19 \quad (\text{in this case})$$

Slide by Andrew Moore

Evaluation on Test Set

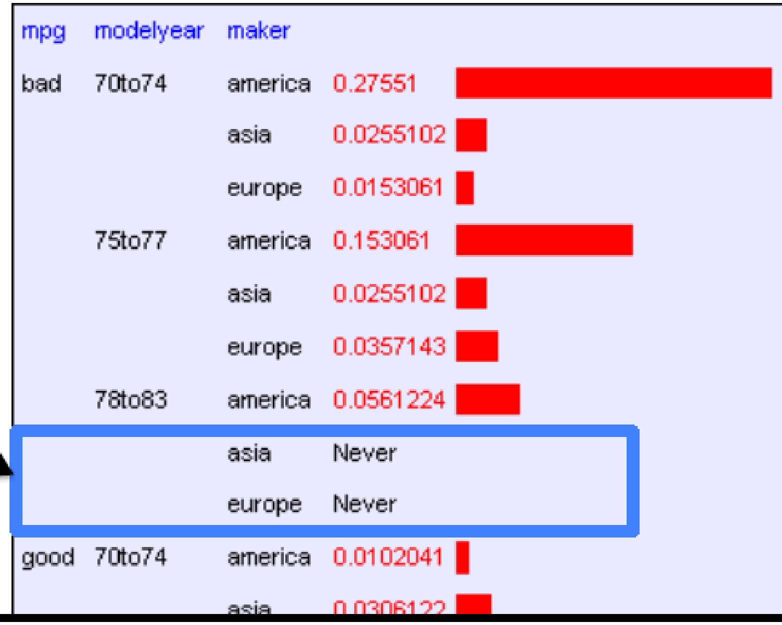
	Set Size	Log likelihood
Training Set	196	-466.1905
Test Set	196	-614.6157

- An independent test set with 196 cars has a much worse log-likelihood
  - Actually it's a billion quintillion quintillion quintillion quintillion times less likely
- Density estimators can overfit...

...and the full joint density estimator is the overfittiest of them all!

Overfitting

If this ever happens, the joint PDE learns there are certain combinations that are impossible

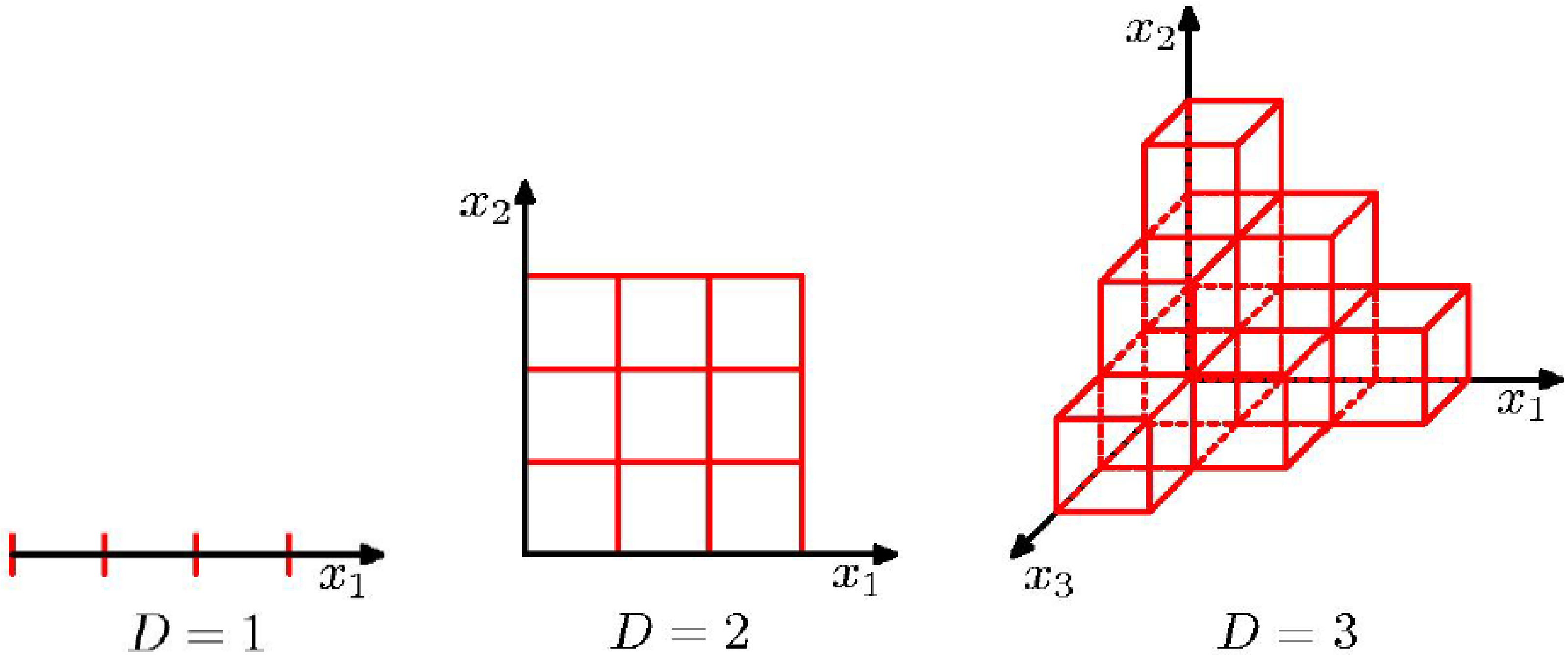


$$\log \hat{P}(\text{dataset} \mid M) = \sum_{i=1}^{n} \log \hat{P}(\mathbf{x}_i \mid M)$$

$$= -\infty \quad \text{if for any } i,\ \hat{P}(\mathbf{x}_i \mid M) = 0$$

de by Andrew Moore

Curse of Dimensionality



Pros and Cons of Density Estimators

- Pros
  - Density Estimators can learn distribution of training data
  - Can compute probability for a record
  - Can do inference (predict likelihood of record)
- Cons
  - Can overfit to the training data and not generalize to test data
  - Curse of dimensionality

Naïve Bayes classifier fixes these cons!

Bayes' Rule

$$P(A \mid B) = \frac{P(B \mid A) \times P(A)}{P(B)}$$

- Exactly the process we just used
- The most important formula in probabilistic machine learning

(Super Easy) Derivation:

$$P(A \wedge B) = P(A \mid B) \times P(B)$$
$$P(B \wedge A) = P(B \mid A) \times P(A)$$

these are the same

Just set equal...

$$P(A \mid B) \times P(B) = P(B \mid A) \times P(A)$$

and solve...

Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London,* **53:370-418**

LDA

- Classify to one of k classes
- Logistic regression computes directly
  - $\mathrm{P}[Y = 1 | X = x]$
  - Assume sigmoid function

Discriminative model

- LDA uses Bayes Theorem to estimate it
  - $\mathrm{P}[Y = k | X = x] = \frac{\mathrm{P}[X = x | Y = k]\mathrm{P}[Y=k]}{\mathrm{P}[X=x]}$
  - Let $\pi_k = \mathrm{P}[Y = k]$ be the prior probability of class k and $f_k(x) = \mathrm{P}[X = x | Y = k]$

Generative model

LDA

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}.$$

Assume $f_k(x)$ is Gaussian!
Unidimensional case (d=1)

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2\right)$$

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_k)^2\right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu_l)^2\right)}.$$

Assumption: $\sigma_1 = \ \ldots \sigma_k = \ \sigma$

Naïve Bayes Classifier

Idea: Use the training data to estimate

$P(X \mid Y)$ and $P(Y)$.

Then, use Bayes rule to infer $P(Y|X_{\text{new}})$ for new data

---

Easy to estimate from data

Impractical, but necessary

$$\mathrm{P}[Y=k|X=x] = \frac{\boxed{\mathrm{P}[Y=k]}\boxed{\mathrm{P}[X_1=x_1 \wedge \cdots \wedge X_d = x_d | Y=k]}}{\boxed{\mathrm{P}[X_1=x_1 \wedge \cdots \wedge X_d = x_d]}}$$

Unnecessary, as it turns out

- Recall that estimating the joint probability distribution $P(X_1, X_2, \ldots, X_d \mid Y)$ is not practical

Naïve Bayes Classifier

Problem: estimating the joint PD or CPD isn't practical

- Severely overfits, as we saw before

However, if we make the assumption that the attributes are independent given the class label, estimation is easy!

$$P(X_1, X_2, \ldots, X_d \mid Y) = \prod_{j=1}^{d} P(X_j \mid Y)$$

- In other words, we assume all attributes are *conditionally independent* given Y
- Often this assumption is violated in practice, but more on that later…

Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

Sky	Temp	Humid	Wind	Water	Forecast	Play?
sunny	warm	normal	strong	warm	same	*yes*
sunny	warm	high	strong	warm	same	*yes*
rainy	cold	high	strong	warm	change	*no*
sunny	warm	high	strong	cool	change	*yes*

$\mathrm{P(play)} = ?$

$\mathrm{P(Sky = sunny \mid play)} = ?$

$\mathrm{P(Humid = high \mid play)} = ?$

$\ldots$

$\mathrm{P(\neg play)} = ?$

$\mathrm{P(Sky = sunny \mid \neg play)} = ?$

$\mathrm{P(Humid = high \mid \neg play)} = ?$

$\ldots$

Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

Sky	**Temp**	**Humid**	**Wind**	**Water**	**Forecast**	**Play?**
sunny	warm	normal	strong	warm	same	*yes*
sunny	warm	high	strong	warm	same	*yes*
rainy	cold	high	strong	warm	change	*no*
sunny	warm	high	strong	cool	change	*yes*

$\mathrm{P(play)} = 3/4$

$\mathrm{P(Sky = sunny \mid play) = ?}$

$\mathrm{P(Humid = high \mid play) = ?}$

...

$\mathrm{P(\neg play)} = 1/4$

$\mathrm{P(Sky = sunny \mid \neg play) = ?}$

$\mathrm{P(Humid = high \mid \neg play) = ?}$

...

Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

Sky	Temp	Humid	Wind	Water	Forecast	Play?
sunny						*yes*
sunny						*yes*
rainy	cold	high	strong	warm	change	*no*
sunny						*yes*

$\mathrm{P(play)} = 3/4$

$\mathrm{P(Sky = sunny \mid play)} = 1$

$\mathrm{P(Humid = high \mid play)} = ?$

$\ldots$

$\mathrm{P(\neg play)} = 1/4$

$\mathrm{P(Sky = sunny \mid \neg play)} = ?$

$\mathrm{P(Humid = high \mid \neg play)} = ?$

$\ldots$

Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

Sky	Temp	Humid	Wind	Water	Forecast	Play?
sunny	warm	normal	strong	warm	same	*yes*
sunny	warm	high	strong	warm	same	*yes*
rainy						*no*
sunny	warm	high	strong	cool	change	*yes*

$\mathrm{P(play)} = 3/4$

$\mathrm{P(Sky = sunny \mid play)} = 1$

$\mathrm{P(Humid = high \mid play)} = ?$

$\ldots$

$\mathrm{P(\neg play)} = 1/4$

$\mathrm{P(Sky = sunny \mid \neg play)} = 0$

$\mathrm{P(Humid = high \mid \neg play)} = ?$

$\ldots$

Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

Sky	Temp	Humid	Wind	Water	Forecast	Play?
		normal				*yes*
		high				*yes*
rainy	cold	high	strong	warm	change	*no*
		high				*yes*

$\mathrm{P(play)} = 3/4$ $\mathrm{P}(\neg\mathrm{play}) = 1/4$

$\mathrm{P(Sky = sunny \mid play)} = 1$ $\mathrm{P}(\mathrm{Sky = sunny} \mid \neg\mathrm{play}) = 0$

$\mathrm{P(Humid = high \mid play)} =$ 2/3 $\mathrm{P}(\mathrm{Humid = high} \mid \neg\mathrm{play}) = ?$

… …

Training Naïve Bayes

Estimate $P(X_j \mid Y)$ and $P(Y)$ directly from the training data by counting!

Sky	Temp	Humid	Wind	Water	Forecast	Play?
sunny	warm	normal	strong	warm	same	*yes*
sunny	warm	high	strong	warm	same	*yes*
		high				*no*
sunny	warm	high	strong	cool	change	*yes*

$\mathrm{P}(\mathrm{play}) = 3/4$

$\mathrm{P}(\mathrm{Sky} = \mathrm{sunny} \mid \mathrm{play}) = 1$

$\mathrm{P}(\mathrm{Humid} = \mathrm{high} \mid \mathrm{play}) = 2/3$

...

$\mathrm{P}(\neg\mathrm{play}) = 1/4$

$\mathrm{P}(\mathrm{Sky} = \mathrm{sunny} \mid \neg\mathrm{play}) = 0$

$\mathrm{P}(\mathrm{Humid} = \mathrm{high} \mid \neg\mathrm{play}) = 1$

...

Laplace Smoothing

- Notice that some probabilities estimated by counting might be zero
  - Possible overfitting!

- Fix by using Laplace smoothing:
  - Adds 1 to each count

$$P(X_j = v \mid Y = k\ ) = \frac{\mathrm{c}_v + 1}{\sum_{v' \in \mathrm{values}(X_j)} \mathrm{c}_{v'} + |\mathrm{values}(X_j)|}$$

where

- c_v is the count of training instances with a value of v for attribute j and class label k
- $|\mathrm{values}(X_j)|$ is the number of values X_j can take on

Using the Naïve Bayes Classifier

- Now, we have

$$\mathrm{P}[Y = k | X = x] = \frac{\mathrm{P}[Y = k]\mathrm{P}[X_1 = x_1 \wedge \cdots \wedge X_d = x_d | Y = k]}{\boxed{\mathrm{P}[X_1 = x_1 \wedge \cdots \wedge X_d = x_d]}}$$

This is constant for a given instance, and so irrelevant to our prediction

– In practice, we use log-probabilities to prevent underflow

- To classify a new point $\mathbf{x}$,

$$h(\mathbf{x}) = \arg\max_{y_k} P(Y = k) \prod_{j=1}^{d} P(X_j = \underbrace{x_j}_{j^{\text{th}} \text{ attribute value of } \mathbf{x}} \mid Y = k)$$

$$= \arg\max_{y_k} \log P(Y = k) + \sum_{j=1}^{d} \log P(X_j = x_j \mid Y = k)$$

Naïve Bayes Classifier

- For each class label k
  1. Estimate prior $P[Y = k]$ from the data
  2. For each value v of attribute X_j
     - Estimate $\mathrm{P}[X_j = v | Y = k]$

- Classify a new point via:

$$h(\mathbf{x}) = \arg\max_{y_k} \log P(Y = k) + \sum_{j=1}^{d} \log P(X_j = x_j \mid Y = k)$$

- In practice, the independence assumption doesn't often hold true, but Naïve Bayes performs very well despite it

Computing Probabilities

- NB classifier gives predictions, not probabilities, because we ignore $\mathrm{P}(\mathrm{X})$ (the denominator in Bayes rule)
- Can produce probabilities by:
  - For each possible class label y_k, compute

$$\underbrace{\tilde{P}(Y = k \mid X = \mathbf{x})}_{\text{This is the numerator of Bayes rule, and is therefore off the true probability by a factor of } \alpha \text{ that makes probabilities sum to 1}} = P(Y = k) \prod_{j=1}^{d} P(X_j = x_j \mid Y = k)$$

  - α is given by $\alpha = \dfrac{1}{\sum_{k=1}^{\#classes} \tilde{P}(Y = k \mid X = \mathbf{x})}$
  - Class probability is given by

$$P(Y = k \mid X = \mathbf{x}) = \alpha \tilde{P}(Y = k \mid X = \mathbf{x})$$

Naïve Bayes Summary

Advantages:

- Fast to train (single scan through data)
- Fast to classify
- Not sensitive to irrelevant features
- Handles real and discrete data
- Handles streaming data well

Disadvantages:

- Assumes independence of features

Document Classification

Test Data:

"planning
language
proof
intelligence"

PROBLEM SETTING

Given:

- Representation of a document
- Set of classes 1,…,K

Determine:

- Class to which document d belongs

(AI) (Programming) (HCI)

Classes:

ML | Planning | Semantics | Garb.Coll. | Multimedia | GUI

Training Data:

ML	Planning	Semantics	Garb.Coll.	Multimedia	GUI
learning	planning	programming	garbage	...	...
intelligence	temporal	semantics	collection		
algorithm	reasoning	language	memory		
reinforcement	plan	proof...	optimization		
network...	language...		region...		

Text Classification: Examples

- Classify news stories as *World, US, Business, SciTech, Sports, etc.*
- Add terms to Medline abstracts (e.g. "Conscious Sedation" [E03.250])
- Classify business names by industry
- Classify student essays as *A/B/C/D/F*
- Classify email as *Spam/Other*
- Classify email to tech staff as *Mac/Windows/ ...*
- Classify pdf files as *ResearchPaper/Other*
- Determine authorship of documents
- Classify movie reviews as *Favorable/Unfavorable/Neutral*
- Classify technical papers as *Interesting/Uninteresting*
- Classify jokes as *Funny/NotFunny*
- Classify websites of companies by Standard Industrial Classification (SIC) code

Bag of Words Representation

What is the ~~best~~ representation for documents?

simplest, yet useful

Idea: Treat each document as a sequence of words

- Assume that word positions are generated *independently*

Dictionary: set of all possible words

- Compute over set of documents
- Use Webster's dictionary, etc.

Bag of Words Representation

Represent document d as a vector of word counts $\mathbf{x}$

- x_j represents the count of word j in the document
  - $\mathbf{x}$ is sparse (few non-zero entries)

number of times "abbey" occurred

aardvark	abacus	abandon	abase	abate	aberration	abbey	abbot	⋮	zoo	
0	0	1	0	0	0	4	0	...	0	$\mathbf{x}$

Another View of Naïve Bayes

- Let the model parameters for class c be given by:

$$\boldsymbol{\theta}_c = \{\theta_{c1}, \theta_{c2}, \ldots, \theta_{c\underbrace{|D|}_{\text{size of dictionary } D}}\}$$

  - $\theta_{cj} = \mathrm{P}($word j occurs in a document from $c)$
  - Also have that $\sum_j \theta_{cj} = 1$

- The likelihood of a document d characterized by $\mathbf{x}$ is

$$P(d \mid \boldsymbol{\theta}_c) = \frac{(\sum_j x_j)!}{\prod_j x_j!} \prod_j (\theta_{cj})^{x_j}$$

  - This is just the multinomial distribution, a generalization of the binomial distribution $\binom{n}{k} p^k (1-p)^{n-k}$

Another View of Naïve Bayes

- The likelihood of a document d characterized by $\mathbf{x}$ is

$$P(d \mid \boldsymbol{\theta}_c) = \frac{(\sum_j x_j)!}{\prod_j x_j!} \prod_j (\theta_{cj})^{x_j}$$

- Use Bayes rule:

introduce class priors

$$\log P(\boldsymbol{\theta}_c \mid d) \propto \log \left(P(\boldsymbol{\theta}_c) \prod_{j=1}^{|D|} (\theta_{cj})^{x_j} \right) = \log P(\boldsymbol{\theta}_c) + \sum_{j=1}^{|D|} x_j \log \theta_{cj}$$

Therefore,

$$h(d) = \arg\max_c \left(\log P(\boldsymbol{\theta}_c) + \sum_{j=1}^{|D|} x_j \log \theta_{cj} \right)$$

This is just a linear decision function!

Document Classification with Naïve Bayes

1. Compute dictionary D over training set (if not given)
2. Represent training documents as bags of words over D
3. Estimate class priors via counting
4. Estimate conditional probabilities as $\hat{\theta}_{cj} = \dfrac{N_{cj} + 1}{N_c + |D|}$
   - N_{cj} is number of times word j occurs in documents from class c
   - N_c is total number of words in all documents from class c

- Naïve Bayes model for new documents (represented in D) is:

$$h(d) = \arg\max_c \left(\log P(c) + \sum_j x_j \hat{w}_{cj} \right)$$

$$\text{where } \hat{w}_{cj} = \log \hat{\theta}_{cj}$$

Review Naïve Bayes

- Density Estimators can estimate joint probability distribution from data
- Risk of overfitting and curse of dimensionality
- Naïve Bayes assumes that features are independent given labels
  - Reduces the complexity of density estimation
  - Even though the assumption is not always true, Naïve Bayes works well in practice
- Applications: text classification with bag-of-words representation
  - Naïve Bayes becomes a linear classifier

Generative model

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
  - Andrew Moore
- Thanks!