DS 4400

Machine Learning and Data Mining I
Spring 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

February 25 2021

Announcements

- Homework 3 is due on Monday, March 8
- Project proposal is due on March 4
- Midterm exam is on Tuesday, March 2
  – During class on Gradescope, 11:45am:1:45pm
  – Short review today

Outline

- Midterm exam review
- Density Estimation
- Naïve Bayes
  - Independence assumption
  - Training Naïve Bayes
  - Laplace smoothing

Midterm Exam Review

What we covered: I

- Bias-Variance Tradeoff
- Linear Regression
  - Closed form simple and multiple Linear Regression
  - Correlation and regression
- Gradient Descent (GD)
  - General algorithm
  - GD for Linear Regression; comparison to closed form
- Non-linear regression: polynomial, spline regression
- Regularization
  - Ridge and Lasso regularization
  - GD for Ridge regression

What we covered: II

- Classifiers
  - Linear vs non-linear classification
  - Generative vs Discriminative models
- kNN classifier
- Logistic regression
  - Maximum Likelihood Estimation (MLE)
  - Cross-entropy objective
  - GD for logistic regression
- Linear Discriminant Analysis (LDA)
- Cross-validation
- Evaluation of classifiers
  - Metrics: precision, recall, F1 score, accuracy, error, confusion matrix
  - ROC curves, AUC

ML Models

- Categorization
  - Is it a linear or non-linear?
  - Is it generative or discriminative?
- For each ML model
  - Understand how training is done
  - Take a small example and train a model
    - E.g., linear regression, LDA
  - Once you have a model know how to evaluate a point and generate a prediction
    - Example: predict probability by logistic regression model or kNN

How to measure performance

- Regression: MSE
- Why we need multiple metrics
  - Accuracy, error
  - Precision, recall
  - Confusion matrix
  - F1 score
  - ROC curves, AUC
- Compute these metrics on small examples

Type I: Conceptual

- Example 1: Describe difference between classification and regression
- Example 2: List two methods for regularization and compare them
- Example 3: Provide advantages and disadvantages, and compare the following:
  - Linear regression with polynomial regression
  - Gradient descent vs closed form solution for linear regression
  - Generative vs discriminative models

Type II: Computational

- Example 1: Given a small dataset, train a particular ML model
  - E.g., linear regression, LDA, etc.
  - Evaluate model on some small training and testing data
- Example 2: Given a particular model, describe the training process and count the number of parameters
- Example 3: Compute different metrics: accuracy, precision, recall, etc.

Type III: Case Study

- Example: Consider the problem of predicting a patient's risk to a disease. The features include demographic information (address, zip code), as well as measurements from blood test results in the last 2 years. Assume there is a datasets including patients with and without the disease.

Describe the process to:

1. Represent the features in a format suitable for ML
2. How would you do feature selection
3. Describe what models you would use and why

Generative vs Discriminative

- Generative model
  - Given X and Y, learns the joint probability $P(X, Y)$
  - Can generate more examples from distribution
  - Examples: LDA, Naïve Bayes, language models (GPT-2, GPT-3, BERT)

- Discriminative model
  - Given X and Y, learns a decision function for classification
  - Examples: logistic regression, kNN

LDA

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

LDA Training and Testing

Given training data $(x_i, y_i), i = 1, \dots, n, y_i \in \{1, \dots, K\}$

1. Estimate mean and variance

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$$

2. Estimate prior

$$\hat{\pi}_k = n_k/n.$$

Given testing point x, predict k that maximizes:

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

Essential probability concepts

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

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

- The joint probability is given by:

$\mathrm{P}(\mathrm{Alarm},$ Theft $) =$

	alarm	¬alarm
theft	0.09	0.01
¬theft	0.1	0.8

Prior probability of theft

$\mathrm{P}($theft$) = 0.1$

by marginalization over Alarm

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

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

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

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

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

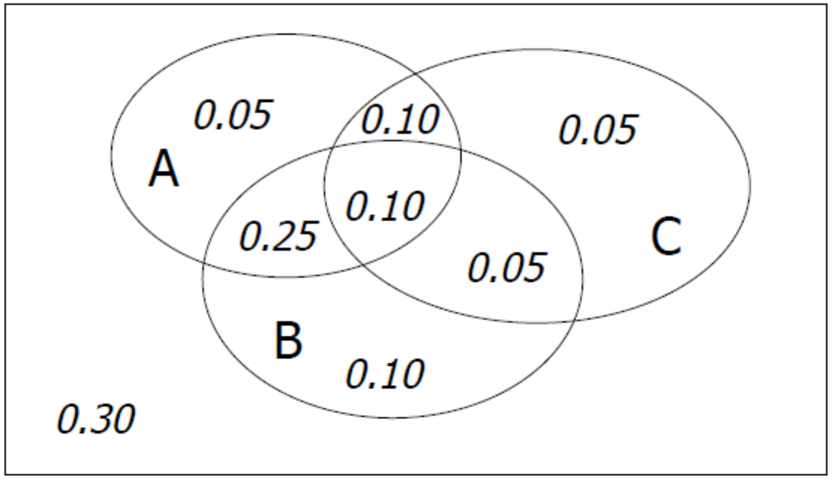
The Joint Distribution

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:

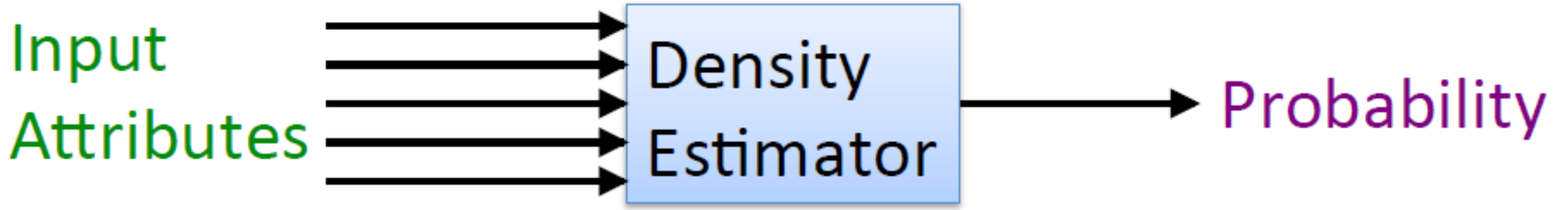
$$\hat{P}(\text{row}) = \frac{\text{records matching row}}{\text{total number of records}}$$

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

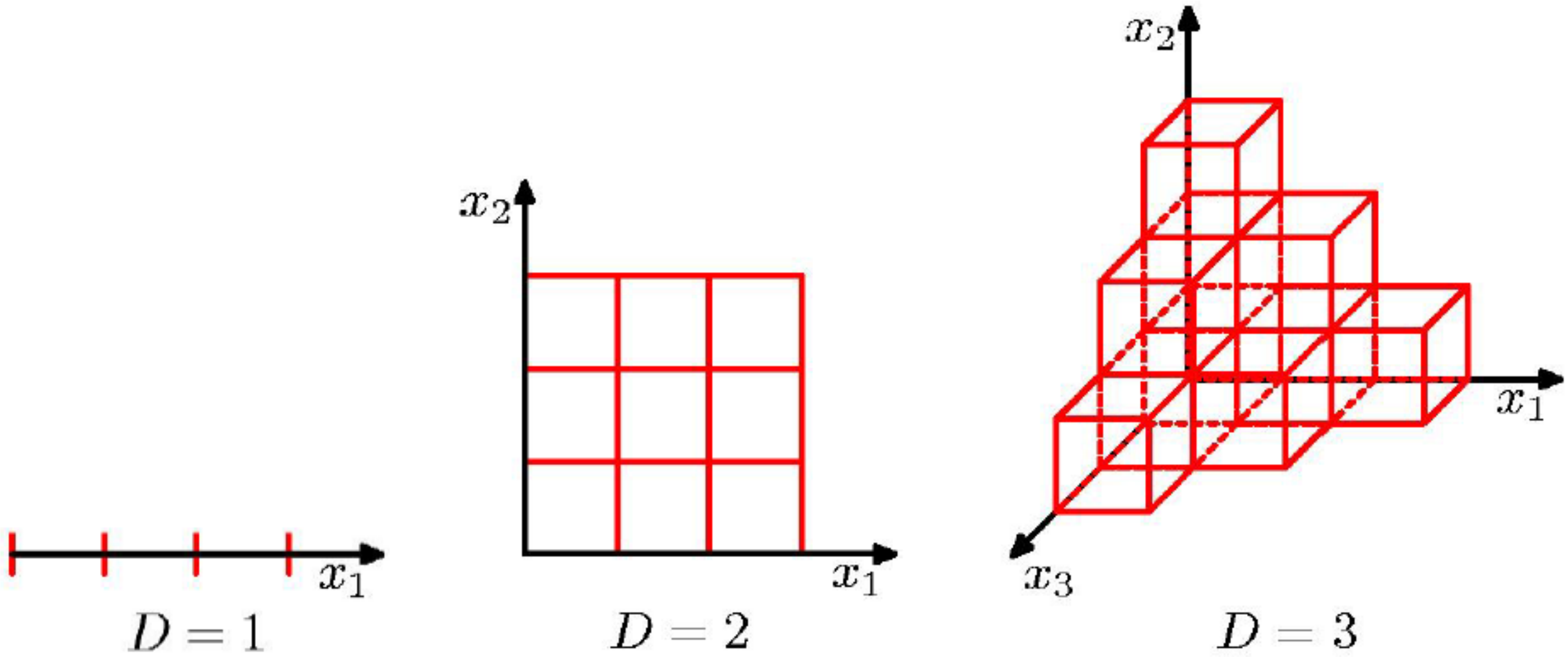
Fraction of all records in which A and B are true but C is false

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



Curse of Dimensionality



Naïve Bayes Classifier

Idea: Use the training data to estimate

$$P(X \mid Y) \text{ and } P(Y).$$

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!

$$\mathrm{P}[X_1 = x_1 \wedge \cdots \wedge X_d = x_d | Y = k] = \prod_{j=1}^{d} P[X_j = x_j | Y = k]$$

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

Using the Naïve Bayes Classifier

- Now, we have

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

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

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 $\pi_k = P[Y = k]$ 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

Training Naïve Bayes

Estimate $P[X_j = x_j | Y = k]$ and $P[Y = k]$ 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*

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

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

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

$\ldots$

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

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

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

$\ldots$

Training Naïve Bayes

Estimate $P[X_j = x_j | Y = k]$ and $P[Y = k]$ 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*

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

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

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

...

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

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

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

...

Training Naïve Bayes

Estimate $P[X_j = x_j | Y = k]$ and $P[Y = k]$ 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*

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

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

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

$\ldots$

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

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

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

$\ldots$

Training Naïve Bayes

Estimate $P[X_j = x_j | Y = k]$ and $P[Y = k]$ directly from the training data by counting!

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

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

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

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

$\ldots$

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

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

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

$\ldots$

Training Naïve Bayes

Estimate $P\left[X_j = x_j \middle| Y = k\right]$ and $P[Y = k]$ 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*

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

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

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

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

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

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

...

...

Training Naïve Bayes

Estimate $P[X_j = x_j | Y = k]$ and $P[Y = k]$ 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*

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

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

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

...

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

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

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

...

Laplace Smoothing

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

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

$$P(X_j = v \mid Y = k) = \frac{c_v + 1}{\sum_{v' \in \text{values}(X_j)} c_{v'} + |\text{values}(X_j)|}$$

where

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

Naïve Bayes Classifier

- For each class label k
  1. Estimate prior $\pi_k = P[Y = k]$ 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

Continuous Features

- Naïve Bayes can be extended to continuous features
- Gaussian Naïve Bayes
  - Here an additional assumption is that each distribution $P\left[X_j|Y=k\right]$ is Gaussian $N(\mu_j, \sigma_j)$
  - It estimates the mean and standard deviation from training data
- This leads to a linear classifier

Comparison to LDA

- Similarity to LDA
  - Both are generative models
  - They both estimate:

    $P[X = x \text{ and } Y = k] = P[X = x|Y = k]P[Y = k]$
- Difference from LDA
  - Naïve Bayes can handle discrete data
  - LDA uses multi-variate normal
  - LDA assumes same variances for all classes
  - Naïve Bayes make the conditional independence assumption

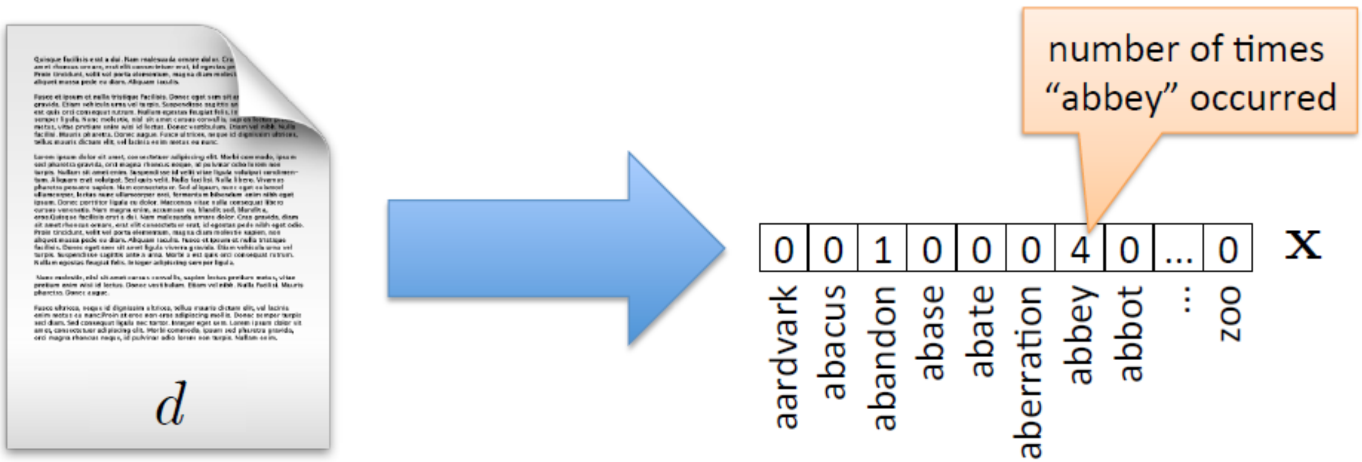
Text Classification: Examples

- 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

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)



- Naïve Bayes learns the distribution of each word per class
- Naïve Bayes becomes a linear classifier under multi-nomial distribution

Naïve Bayes Summary

Advantages:

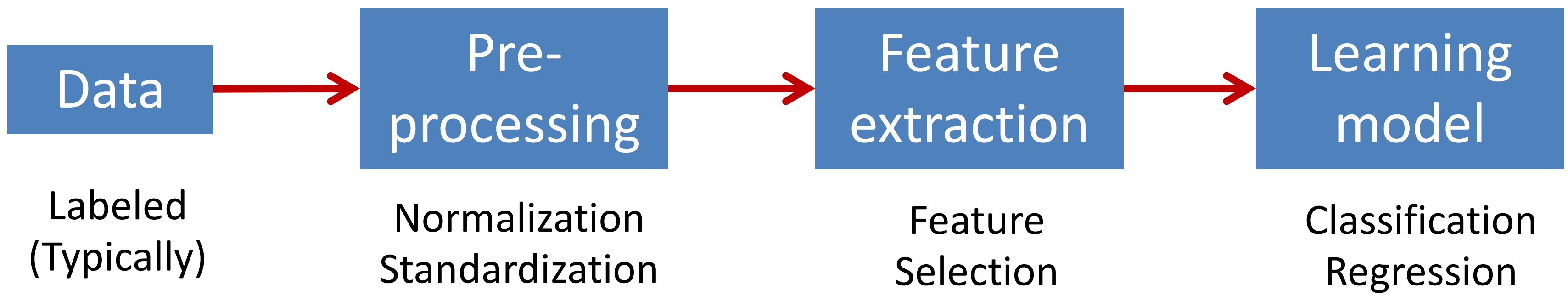
- Fast to train (single scan through data)
- Fast to classify
- Not sensitive to irrelevant features
- Handles real and discrete data
- Handles streaming data well

Disadvantages:

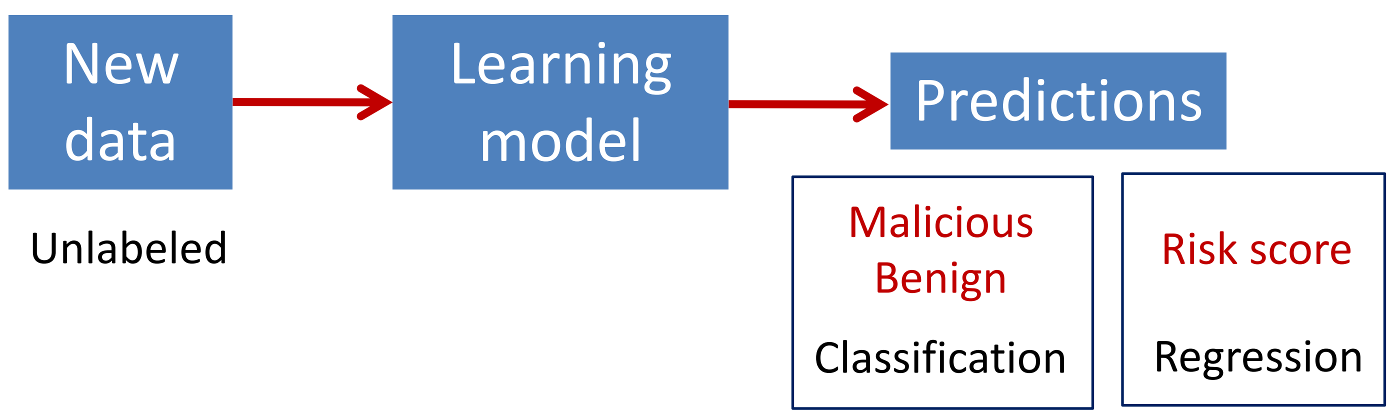
- Assumes independence of features

Supervised Learning Process

Training



Testing



Feature selection

- *Feature Selection*
  - Process for choosing an optimal subset of features according to a certain criteria

- Why we need Feature Selection:
  1. To improve performance (in terms of speed, predictive power, simplicity of the model).
  2. To visualize the data for model selection.
  3. To reduce dimensionality and remove noise.

Methods for Feature Selection

- Wrappers
  - Select subset of features that gives best prediction accuracy (using cross-validation)
  - Model-specific
- Filters
  - Compute some statistical metrics (correlation coefficient, information gain)
  - Select features with statistics higher than threshold
- Embedded methods
  - Feature selection done as part of training
  - Example: Regularization (Lasso, L1 regularization)

Wrappers: Search Strategy

- With an **exhaustive search**

$$101110000001000100001000000000100101010$$

With d features $\rightarrow$ 2^d possible feature subsets.

20 features … 1 million feature sets to check
25 features … 33.5 million sets
30 features … 1.1 billion sets

- Need for **a search strategy**
  - Sequential forward selection
  - Recursive backward elimination
  - Genetic algorithms
  - Simulated annealing
  - …

Wrappers: Sequential Forward Selection

Start with the empty set $S = \emptyset$

While *stopping criteria not met*

For each feature X_f not in S

- Define $S' = S \cup \{X_f\}$
- Train model using the features in S'
- Compute the accuracy on validation set

End

$S = S'$ where S' is the feature set with the greatest accuracy

End

Backward feature selection starts with all features and eliminates backward

Filters

Principle: *replace evaluation of model with quick to compute statistics* $J(X_f)$

k	$J(X_k)$
35	0.846
42	0.811
10	0.810
654	0.611
22	0.443
59	0.388
...	...
212	0.09
39	0.05

For each feature X_f
- Compute $J(X_f)$

End

Rank features according to $J(X_f)$
Choose manual cut-off point

Examples of filtering criterion

- The Information Gain with the target variable $J(X_f) = I(X_f; Y)$
- The correlation with the target variable
- Feature importance

Embedded methods: Regularization

Principle: the classifier performs feature selection as part of the learning procedure

Example: the logistic LASSO (Tibshirani, 1996)

$$f(\boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{w}^T \boldsymbol{x})}} = P(Y = 1|\boldsymbol{x})$$

With Error Function:

$$E = \underbrace{-\sum_{i=1}^{N} \{y_i \log f(\boldsymbol{x}_i) + (1 - y_i) \log(1 - f(\boldsymbol{x}_i))\}}_{\text{Cross-entropy error}} + \underbrace{\lambda \sum_{f=1}^{d} |w_f|}_{\text{Regularizing term}}$$

Pros:

- Performs feature selection as part of learning the procedure

Summary: Feature Selection

- Filtering
- L_1 regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

Acknowledgements

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