

# Supervised Machine Learning - A

# Road Map

- **Basic concepts**

- Regression
- Naïve Bayesian classification
- K-nearest neighbor
- Support vector machines
- Decision tree induction
- Ensemble methods: Bagging and Boosting
- Summary

# What do we mean by learning?

- **Given**

- a data set  $D$ ,
- a task  $T$ , and
- a performance measure  $M$ ,

a computer system is said to **learn** from  $D$  to perform the task  $T$  if after learning the system's performance on  $T$  improves as measured by  $M$ .

- In other words, the learned model helps the system to perform  $T$  better as **compared to no learning**.

## Fundamental assumption of learning

**Assumption:** The distribution of training examples is **identical** to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

# Why Use Machine Learning?

- We have more data than ever before.
- For example, programming a rules-based system for classification could require hundreds of rules. This process only works well if you know all the situations under which decisions can be made.
- For large data sets, machine learning systems are often very efficient at classifying data.

# Overview of Machine Learning

- Machine learning uses statistical techniques, allowing computers to "learn" from data to identify patterns and make predictions.
- Machine learning can be supervised, unsupervised, (semi-supervised, or reinforced).

# Supervised or Unsupervised?

## Machine Learning

```
graph TD; ML[Machine Learning] --> S[Supervised]; ML --> U[Unsupervised];
```

### Supervised

- Predictive model is developed based on labeled training data.

### Unsupervised

- Discovery of an internal pattern or structure is based on unlabeled training data.

# Supervised or Unsupervised? (cont.)

- With supervised machine learning, the model is trained using examples of labeled input-output pairs.
- In other words, the computer is first being shown a set of correctly labeled examples. It creates a predictive model based on those examples.



# Supervised Machine Learning Example

- For example, a set of news articles could be pre-labeled as being either about sports, entertainment, or politics.
- The supervised machine learning algorithm would “learn” from that labeled data to create a predictive model for classifying a new, unlabeled article into one of the three classes.

# Unsupervised Machine Learning Example

- An example of unsupervised machine learning would use unlabeled news articles as the training data. The output classes in this case would not be known in advance.
- The algorithm could then, for example, be used to organize the articles into “clusters” based on similarity of content.

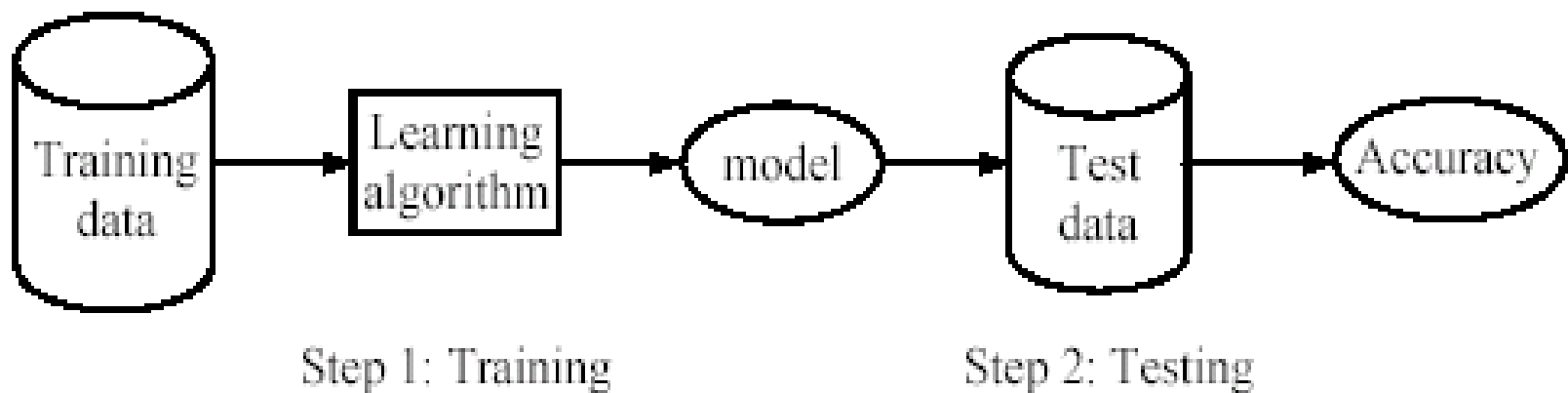
# Supervised vs. unsupervised Learning

- **Supervised learning:** classification is seen as supervised learning from examples.
  - **Supervision:** The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (**supervision**).
  - Test data are classified into these classes too.
- **Unsupervised learning (clustering)**
  - **Class labels of the data are unknown**
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

Supervised learning process: two steps

- **Learning (training)**: Learn a model using the training data
- **Testing**: Test the model using unseen test data to assess the model accuracy or errors

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



# An example application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- **A decision is needed**: whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- **Problem**: to predict **high-risk patients** and discriminate them from **low-risk patients**.

## Another application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant:
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- **Problem**: to decide whether an application should be approved, or to classify applications into two categories, **approved** and **not approved**.

# Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have “experiences”.
- A computer system learns from data, which represent some “past experiences” of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: Supervised learning, classification, or inductive learning.

## An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?



# An example: test data (loan application)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

# Road Map

- Basic concepts
- **Regression**
- Naïve Bayesian classification
- K-nearest neighbor
- Support vector machines
- Decision tree induction
- Ensemble methods: Bagging and Boosting
- Summary

# Regression

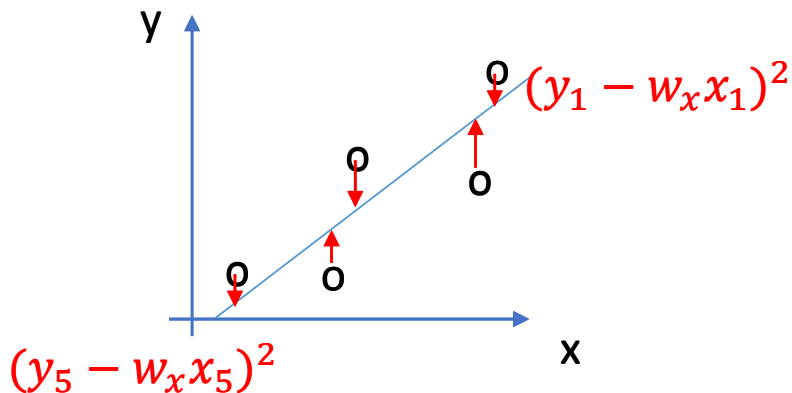
Residual sum of square is given

$$\text{RSS}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y_n - \mathbf{w}^\top \mathbf{x}_n)^2$$

$$(y_n - \mathbf{w}^\top \mathbf{x}_n)^2$$

$$[y_1 \quad \cdots \quad y_n] - [w_x] [x_1 \quad \cdots \quad x_n]$$

$$= [(y_1 - w_x x_1)^2 \quad \cdots \quad (y_n - w_x x_n)^2]$$



Residual sum of square is given

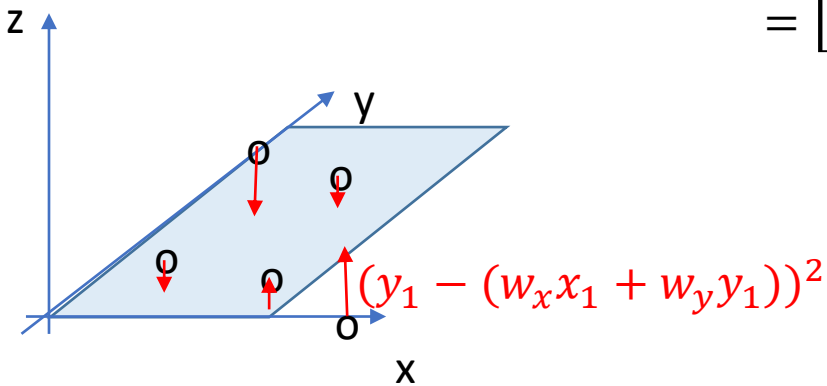
$$\text{RSS}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N (y_n - \mathbf{w}^T \mathbf{x}_n)^2$$

$$(y_n - \mathbf{w}^T \mathbf{x}_n)^2$$

$$[y_1 \quad \cdots \quad y_n] - \begin{bmatrix} w_x \\ w_y \end{bmatrix}^T [x_1 \quad \cdots \quad x_n]$$

Add one dimension

$$= [(y_1 - (w_x x_1 + w_y y_1))^2 \quad \cdots \quad (y_n - (w_x x_n + w_y y_n))^2]$$



# Multi-Regression and Polynomial Regression

- The simplest non-linear model we can consider, for a response  $Y$  and a predictor  $X$ , is a polynomial model of degree  $M$ ,

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \dots + \beta_M x^M + \epsilon.$$

- Just as in the case of linear regression with cross terms, polynomial regression is a special case of linear regression - we treat each  $x^m$  as a separate predictor. Thus, we can write

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \dots & x_1^M \\ 1 & x_2^1 & \dots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^M \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{pmatrix}.$$

# Polynomial Regression

## Multi-Regression

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,J} \\ 1 & x_{2,1} & \dots & x_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,J} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_J \end{pmatrix},$$

## Poly-Regression

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \dots & x_1^M \\ 1 & x_2^1 & \dots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n^1 & \dots & x_n^M \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{pmatrix}.$$

# Model Selection

- **Model selection** is the application of a principled method to determine the complexity of the model, e.g. choosing a subset of predictors, choosing the degree of the polynomial model etc.
- There are too many predictors:
  - the feature space has high dimensionality
  - the polynomial degree is too high
  - too many cross terms are considered

# Model Selection

- **Question:**

- How many different models when considering  $J$  predictors?



# Model Selection

- **Example: 3 predictors ( $X_1, X_2, X_3$ )**

- Models with 0 predictor:

- 

- Models with 1 predictor:

- 

- 

- 

- Models with 2 predictors:

- 

- 

- 

- Models with 3 predictors:

- 

M0:

M1:  $X_1$

M2:  $X_2$

M3:  $X_3$

M4:  $\{X_1, X_2\}$

M5:  $\{X_2, X_3\}$

M6:  $\{X_3, X_1\}$

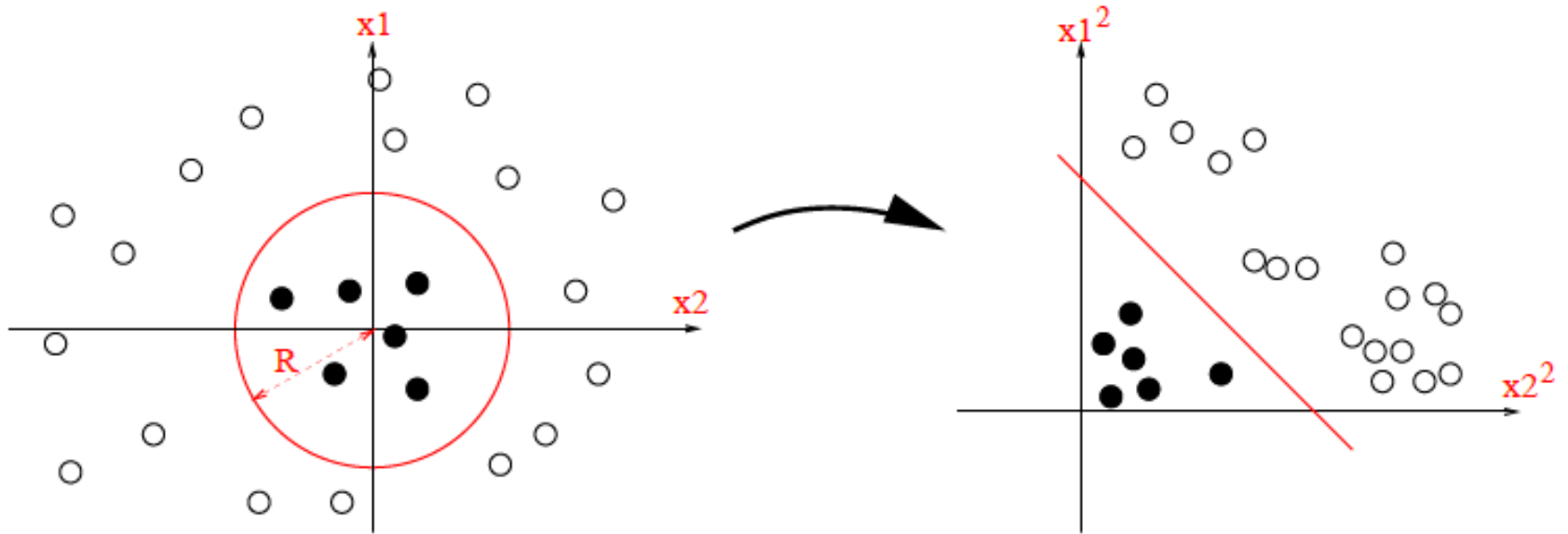
M7:  $\{X_1, X_2, X_3\}$

$2^J$  Models

# Stepwise Variable Selection and Cross Validation

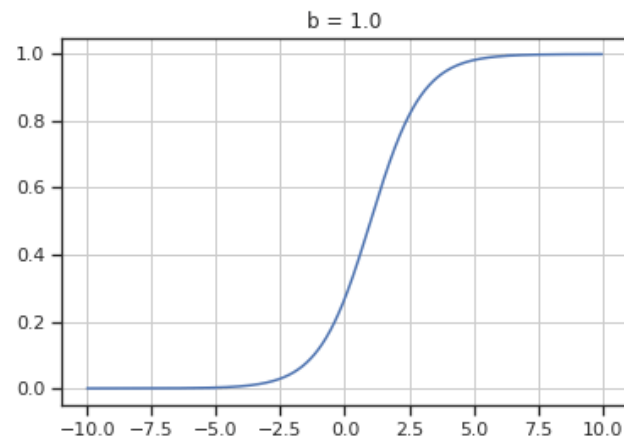
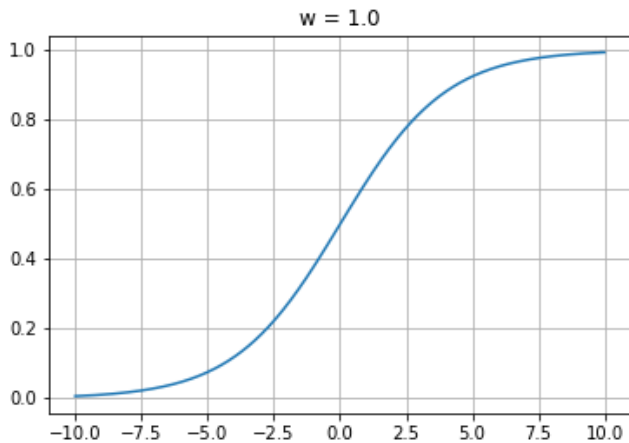
- Selecting optimal subsets of predictors (including choosing the degree of polynomial models) through:
- stepwise variable selection - iteratively building an optimal subset of predictors by optimizing a fixed model evaluation metric each time,
- validation - selecting an optimal model by evaluating each model on validation set.

# Logistic regression



## Sigmoid function for logistic regression

***b*** and ***w*** determine the steepness of the sigmoid function



# Road Map

- Basic concepts
- Regression
- **Naïve Bayesian classification**
- K-nearest neighbor
- Support vector machines
- Decision tree induction
- Ensemble methods: Bagging and Boosting
- Summary

# Bayesian classification

- **Probabilistic view:** Supervised learning can naturally be studied from a probabilistic point of view.
- Let  $A_1$  through  $A_k$  be attributes with discrete values. The class is  $C$ .
- Given a test example  $d$  with observed attribute values  $a_1$  through  $a_k$ .
- Classification is basically to compute the following posteriori probability. The prediction is the class  $c_j$  such that

$$\Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|})$$

is maximal

# Naïve Bayesian classifier

- Advantages:
  - Easy to implement
  - Very efficient
  - Good results obtained in many applications
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy when the assumption is seriously violated (those highly correlated data sets)

## Discussions


- Most assumptions made by naïve Bayesian learning are violated to some degree in practice.
- Despite such violations, researchers have shown that naïve Bayesian learning produces very accurate models.
- Naïve Bayesian learning is extremely efficient.



# Using a Naive Bayes Classifier

- To create our classification tool, we used Scikit-learn, a free software machine learning library for the Python programming language.
- Our classification tool uses the multinomial naive Bayes algorithm.
- There are many other potential algorithms you could use for classification.
- For simplicity and effectiveness, naive Bayes is a good candidate for an introduction to machine learning.

# Bayes' Theorem

$$P(A | B) = \frac{P(B|A) P(A)}{P(B)}$$
The diagram features two arrows. One arrow originates from the bottom-left text box and points diagonally upwards to the term  $P(A | B)$  in the equation. The second arrow originates from the bottom-right text box and points diagonally upwards to the term  $P(B)$  in the denominator of the equation.

The probability of event A, given B.

The probability of event B, given A, multiplied by the prior probability of A, divided by the prior probability of B.

Using Bayesian probability, you reason backwards to find out the events—or random variables—that most likely led to a specific outcome.

# Naive Bayes Assumption

- The naive Bayes algorithm is based on Bayes' theorem of conditional probability, applied with the “naive” assumption that there is independence between features.
- In other words, the effect of the value of a predictor ( $x$ ) on a given class ( $c$ ) is independent of the values of other predictors.
- Is the occurrence of a particular feature really independent of the occurrence of other features?

# Naive Bayes Classification Example

- For the purposes of illustrating how a naive Bayes classifier works, imagine five extremely short email documents, each only a few words long.
- In this example, the classification problem is to classify the email as being either spam (unsolicited anonymous bulk advertising) or not spam.

# Naive Bayes Classification (cont.)

- An annotator has already labeled each of the five emails as being either spam or not spam. This is considered our training set of data.

Class

**Doc 1** Follow-up meeting

Not Spam

**Doc 2** Free cash. Get money.

SPAM

**Doc 3** Money! Money! Money!

SPAM

**Doc 4** Dinner plans

Not Spam

**Doc 5** GET CASH NOW

SPAM

# Naive Bayes Classification (cont.)

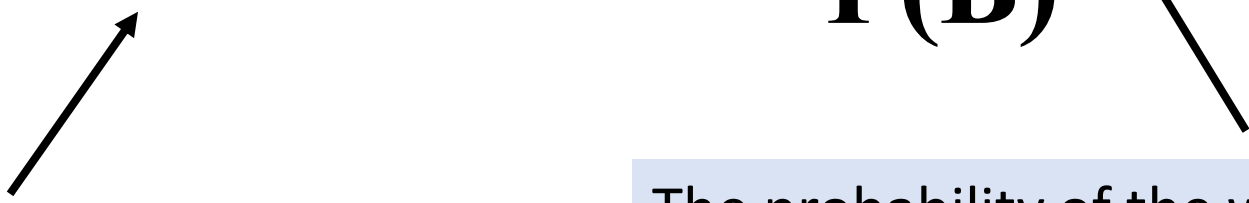
- Consider the following new, unclassified example:

**Doc 6** Get Money Now

Spam or Not Spam?

- The naive Bayes algorithm will calculate the probability, based on the previous results from the training data, that Doc 6 is spam. It will also calculate the probability that it is not spam.

# Applying Bayes' Theorem

$$P(A | B) = \frac{P(B|A) P(A)}{P(B)}$$
The diagram consists of two arrows. One arrow originates from the bottom-left text box and points diagonally upwards to the term  $P(A | B)$  in the equation. The other arrow originates from the bottom-right text box and points diagonally upwards to the term  $P(B)$  in the denominator of the equation.

The probability of an email being spam, given the words in the email.

The probability of the words in the email, given they appeared in spam, multiplied by the prior probability of the email being spam, divided by the prior probability of the words used in the email.

# Prior Probability

- First, we need to calculate the prior probability of each class.
- In other words, in our training set, what was the overall probability that a document was spam? What was the overall probability that a document was not spam?

Doc 1	Not Spam
Doc 2	SPAM
Doc 3	SPAM
Doc 4	Not Spam
Doc 5	SPAM


$$\text{Spam} = 3/5 = .6$$

$$\text{Not Spam} = 2/5 = .4$$




# Naive Bayes Classification

$$P(A | B) = \frac{P(B|A) P(A)}{P(B)}$$

$$P(\text{spam} | \text{get money now}) = \frac{P(\text{get money now} | \text{spam}) P(\text{spam})}{P(\text{get money now})}$$


.6

$$P(\text{not spam} | \text{get money now}) = \frac{P(\text{get money now} | \text{not spam}) P(\text{not spam})}{P(\text{get money now})}$$


.4

# Naive Bayes Classification (cont.)

- To classify the text as spam or not spam, we will look for the label having the bigger probability. Therefore, we can eliminate the divisor, which is the same for both labels.

$$P(\text{spam} | \text{get money now}) = \frac{P(\text{get money now} | \text{spam})}{P(\text{get money now})} (.6)$$

~~$P(\text{get money now})$~~

$$P(\text{not spam} | \text{get money now}) = \frac{P(\text{get money now} | \text{not spam})}{P(\text{get money now})} (.4)$$

~~$P(\text{get money now})$~~

# Naive Bayes Classification (cont.)

- Recall, in naive Bayes each feature is seen as being independent.
- In other words, each word is calculated as having a unique probability:

$$P(\text{get money now}) = P(\text{get}) \times P(\text{money}) \times P(\text{now})$$

# Word Frequency Table

Word	Not Spam	Spam
follow-up	1	0
meeting	1	0
free	0	1
cash	0	2
money	0	4
dinner	1	0
plans	1	0
get	0	2
now	0	1
Total	4	9

- How many times each word occurs in each class is counted in the training data.

# Calculating Conditional Probabilities

Word	Not Spam	Spam
follow-up	$1/4 = .25$	$0/9 = .00$
meeting	$1/4 = .25$	$0/9 = .00$
free	$0/4 = .00$	$1/9 = .11$
cash	$0/4 = .00$	$2/9 = .22$
money	$0/4 = .00$	$4/9 = .44$
dinner	$1/4 = .25$	$0/9 = .00$
plans	$1/4 = .25$	$0/9 = .00$
get	$0/4 = .00$	$2/9 = .22$
now	$0/4 = .00$	$1/9 = .11$
Total	4	9

- The conditional probability of each word occurring in a given class is calculated.
- However, what happens when words do not appear in a class of the training data?

# Avoiding Zeros in the Calculation

Word	Not Spam	Spam
follow-up	$1/4 = .25$	$0/9 = .00$
meeting	$1/4 = .25$	$0/9 = .00$
free	$0/4 = .00$	$1/9 = .11$
cash	$0/4 = .00$	$2/9 = .22$
money	$0/4 = .00$	$4/9 = .44$
dinner	$1/4 = .25$	$0/9 = .00$
plans	$1/4 = .25$	$0/9 = .00$
get	$0/4 = .00$	$2/9 = .22$
now	$0/4 = .00$	$1/9 = .11$
Total	4	9

- Words not occurring have a conditional probability of 0.
- Multiplying by zero would nullify the naive Bayes calculation. We need to smooth the data.

# Laplace Smoothing

Word	Not Spam	Spam
follow-up	1 + 1	0 + 1
meeting	1 + 1	0 + 1
free	0 + 1	1 + 1
cash	0 + 1	2 + 1
money	0 + 1	4 + 1
dinner	1 + 1	0 + 1
plans	1 + 1	0 + 1
get	0 + 1	2 + 1
now	0 + 1	1 + 1
Total	4 + 9 = 13	9 + 9 = 18

- Using Laplace smoothing, we go back, adding 1 to every word count.
- For balance, we add the number of all unique possible words (total vocabulary) to the divisor, so the division will never be greater than 1.

# Class Conditional Probabilities

Word	Not Spam	Spam
follow-up	$2/13 = .153$	$1/18 = .055$
meeting	$2/13 = .153$	$1/18 = .055$
free	$1/13 = .077$	$2/18 = .111$
cash	$1/13 = .077$	$3/18 = .167$
money	$1/13 = .077$	$5/18 = .278$
dinner	$2/13 = .153$	$1/18 = .055$
plans	$2/13 = .153$	$1/18 = .055$
get	$1/13 = .077$	$3/18 = .167$
now	$1/13 = .077$	$2/18 = .111$
Total	$4 + 9 = 13$	$9 + 9 = 18$

- Now that we have smoothed the data, the conditional probability of each word occurring in a given class is recalculated.



# Naive Bayes Classification

$$P(\text{spam} \mid \text{get money now}) = P(\text{get}) \times P(\text{money}) \times P(\text{now}) \times (.6)$$

$$P(\text{spam} \mid \text{get money now}) = P(.167) \times P(.278) \times P(.111) \times (.6) = .0031$$

$$P(\text{not spam} \mid \text{get money now}) = P(\text{get}) \times P(\text{money}) \times P(\text{now}) \times (.4)$$

$$P(\text{not spam} \mid \text{get money now}) = P(.077) \times P(.077) \times P(.077) \times (.4) = .0002$$

- We calculate the probability of an email containing the words “get money now” in each of the two classes.

# Naive Bayes Classification (cont.)

$$P(\text{spam} \mid \text{get money now}) = P(\text{get}) \times P(\text{money}) \times P(\text{now}) \times (.6)$$

$$P(\text{spam} \mid \text{get money now}) = P(.167) \times P(.278) \times P(.111) \times (.6) = .0031$$

$$P(\text{not spam} \mid \text{get money now}) = P(\text{get}) \times P(\text{money}) \times P(\text{now}) \times (.4)$$

$$P(\text{not spam} \mid \text{get money now}) = P(.077) \times P(.077) \times P(.077) \times (.4) = .0002$$

- A higher value indicates a higher probability. In this case, there is a higher probability of these words occurring in an email that is spam. The classifier would therefore label the email as spam.

# Features for Spam Classification

This was a simplified example. In reality, spam detection systems use more than just individual words to classify email as spam or not spam. Spam classifiers use many additional features, including:

- Sender's address
- IP address
- Use of capitalization
- Specific phrases
- Whether or not the text contains a link