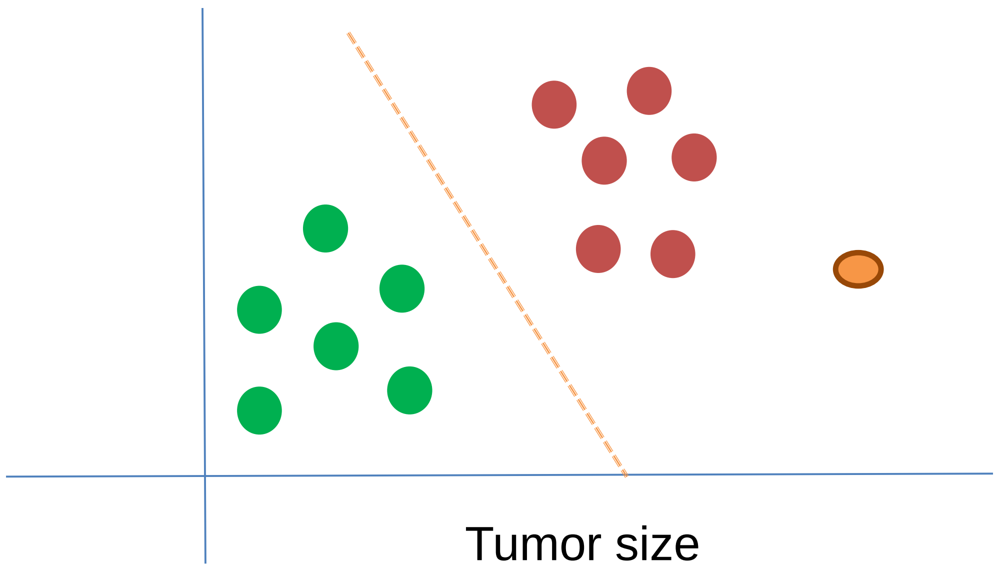
NAÏVE BAYES

Generative vs. Discriminative Classifiers

Training classifiers involves estimating f: X → Y, or P(Y|X)

Discriminative classifiers (also called ‘informative’ by Rubinstein&Hastie):

1. Assume some functional form for P(Y|X)
2. Estimate parameters of P(Y|X) directly from training data

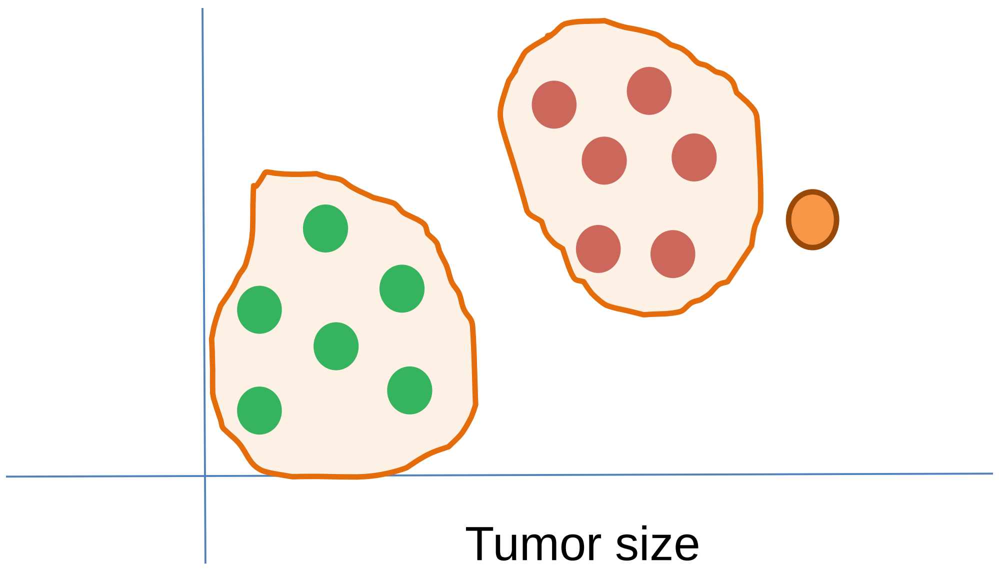


Generative vs. Discriminative Classifiers

Training classifiers involves estimating f: X → Y, or P(Y|X)

Generative classifiers

1. Assume some functional form for P(X|Y), P(X)
2. Estimate parameters of P(X|Y), P(X) directly from training data
3. Model predictions based on P(Y|X= x_i)



Maximum A Posteriori (MAP) Classifier

- Given the data feature vector ***x***, we would like to find the class with the largest probability:

$$\hat{C} \triangleq \arg\max_{C} p(C|\boldsymbol{x})$$

- To accomplish this, we iterate through all possible classes C_1, C_2, …., C_K and evaluate the quantity $p(C_i|\boldsymbol{x})$, and pick the class that has the largest probability.

MAP Classifier: Bayes Rule

- How do we compute $p(C_i|\boldsymbol{x})$? Apply Bayes' Rule!

likelihood prior

$$p(A|B)p(B) = p(B|A)p(A) \qquad \Rightarrow p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

evidence

- Applying Bayes' rule to $p(C_i|\boldsymbol{x})$, we obtain:

$$p(C_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_i)p(C_i)}{p(\boldsymbol{x})}$$

MAP Classifier: Bayes Rule (2)

$$p(C_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_i)p(C_i)}{p(\boldsymbol{x})}$$

- $p(C_i)$ is called the prior probability of a class
- $p(\boldsymbol{x}|C_i)$ is called the likelihood of the data (what is the probability of observing ***x*** if the class was C_i)
- p(x) is the probability of seeing the data so its simply a normalizing factor (applies to all C_i), so doesn't affect which C_i attains MAP.

MAP Classifier: Bayes Rule (3)

- To compute the normalizing factor, we use:

$$p(\boldsymbol{x}) = \sum_{i=1}^{K} p(\boldsymbol{x}|C_i)p(C_i)$$

So that

$$\sum_{i=1}^{K} p(C_i|\boldsymbol{x}) = 1$$

Hence,

$$p(C_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_i)p(C_i)}{\sum_{\ell=1}^{K} p(\boldsymbol{x}|C_\ell)p(C_\ell)}$$

But really, we don't need to calculate the denominator p(x), so we can simply write it as

$$p(C_i|x) = p(x|C_i)p(C_i)$$

MAP Classifier: Bayes Rule (4)

- Usually, the likelihood $p(\boldsymbol{x}|C_i)$ is difficult to compute because it is N-dimensional (length of feature vector).
- This is because the distribution considers correlations between the features when computing the likelihood.

- We can write $p(\boldsymbol{x}|C_i)$ equivalently as:

$$p(\boldsymbol{x}|C_i) = p(x_1, x_2, \ldots, x_N|C_i)$$

which makes the dependence on individual features explicit.

Naïve Bayes Classifier: Independence Assumption

- The Naïve Bayes classifier introduces one major assumption regarding the features: independence
- That is, the Naïve Bayes classifier assumes:

$$p(\boldsymbol{x}|C_i) = p(x_1, x_2, \ldots, x_N|C_i) = \prod_{n=1}^{N} p(x_n|C_i)$$

- That is, the complicated likelihood $p(\boldsymbol{x}|C_i)$ can now be factored into a product of N 1-dimensional likelihoods, which are easy to compute.

- It is good to note that independence implies that the features are not correlated (but generally not the other way around---so independence is a stronger assumption).

Naïve Bayes Classifier

- With the independence assumption, the MAP classifier (now called the Naïve Bayes Classifier) can be written as:

$$p(C_i|\boldsymbol{x}) = \frac{\prod_{n=1}^{N} p(x_n|C_i)p(C_i)}{\sum_{\ell=1}^{K} \prod_{n=1}^{N} p(x_n|C_\ell)p(C_\ell)}$$

- It turns out that the Naïve Bayes classifier works very well on empirical datasets even if the independence assumption doesn't actually hold.

Naïve Bayes Classifier

$$p(C_i|\boldsymbol{x}) = \frac{\prod_{n=1}^{N} p(x_n|C_i)p(C_i)}{}$$

- What happens our training example does not have an instance that is present in the test data?

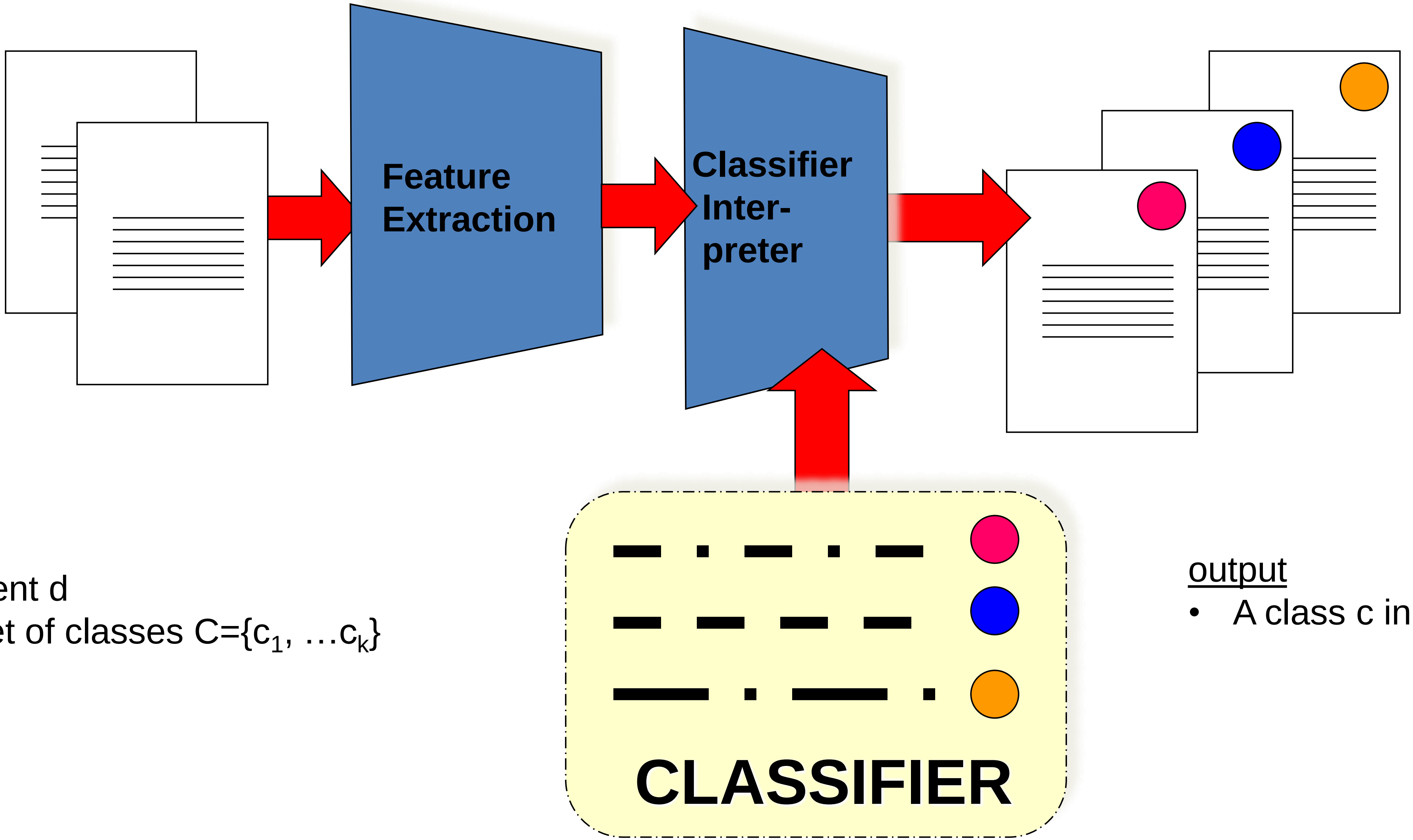
- To eliminate zeros, we use add-one or Laplace smoothing, which simply adds one to each count.

Now, let us see Spark documentation: http://spark.apache.org/docs/latest/mllib-naive-bayes.html

EXAMPLES OF TEXT CATEGORIZATION

- Spam Detection
  - “spam” / “not spam”
- Assigning subject categories, topics, or genres
  - “finance” / “sports” / “asia”
- Sentiment analysis
  - “like” / “hate” / “neutral”
- Authorship iden1fica1on
  - “Shakespeare” / “Marlowe” / “Ben Jonson”
  - The Federalist papers

TEXT CATEGORIZATION



Input

- A document d
- A fixed set of classes C={c_1, …c_k}

output

- A class c in C

Text Classification Algorithms: Learning

- From training corpus, extract Vocabulary
- Calculate required $P(c_j)$ and $P(x_k \mid c_j)$ terms
  - For each c_j in C do
    - $docs_j \leftarrow$ subset of documents for which the target class is c_j

$$P(c_j) \Leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

    - $Text_j \leftarrow$ single document containing all $docs_j$
      - for each word x_k in *Vocabulary*
        - $n_k \leftarrow$ number of occurrences of x_k in $Text_j$

$$P(x_k \mid c_j) \Leftarrow \frac{n_k + 1}{n + |Vocabulary|}$$

This is called the Bag of Words model…. But how do we select this vocabulary??

Can't possibly use all words!

Chi Square, TF/IDF, etc.

Text Classification Algorithms: Classifying

- positions ← all word positions in current document which contain tokens found in Vocabulary

- Return cNB, where

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Naïve Bayes Posterior Probabilities

- Classification results of naïve Bayes (the class with maximum posterior probability) are usually fairly accurate.

- However, due to the inadequacy of the conditional independence assumption, the actual posterior-probability numerical estimates are not.
  - Output probabilities are generally very close to 0 or 1.