Week3

Text classification - Supervised learning

Examples: Topic identification, Spam detection. Sentiment analysis, Spelling correction

Feature identification

Types of textual features:

- Words
    - By far the most common feature.
    - Requires handling of common words: stop words (*the, of, a)*.
    - Normalization: make lower case vs leave as-is.
    - Stemming / Lemmatization.
- Characteristics of words: Capitalization (White House vs white house).
- Parts of Speech (POS) of words in a sentence.

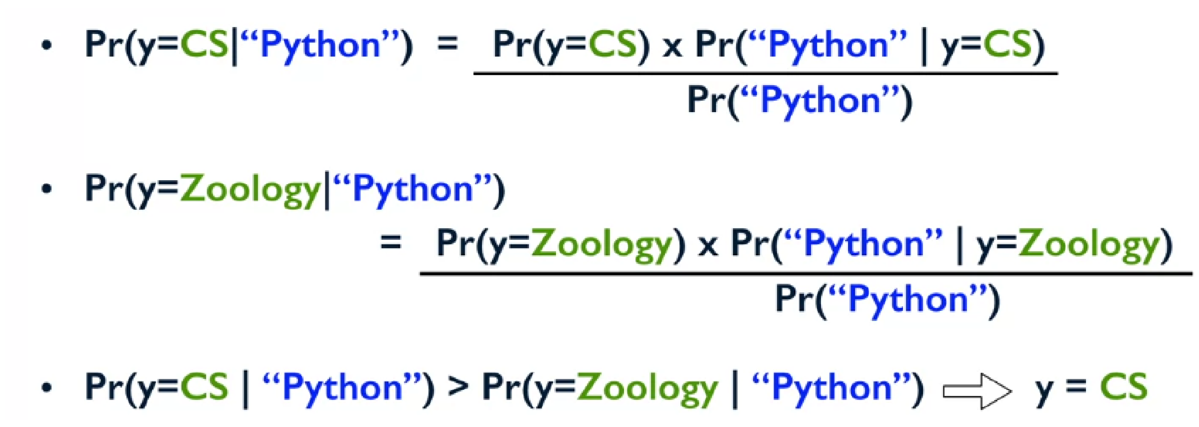
- Grammatical structure, sentence parsing.
- Grouping words of similar meaning or semantics to the same feature(s): {buy, purchase}, {Mr, Ms, Dr, Prof}, Numbers/Digits, Dates.
- Depending on the task, features may come from inside words and sentences
    - bigrams, trigrams, n-grams: “White House”.
    - character sub-sequences in words: “ing”, “ion”, …

Naive Bayes classifier: Probabilistic model

It doesn't give context or associate words. The probability of White followed by House is the same as White cat → All features are independent.

Suppose we want to classify the word '*python*' given 3 classes. And in general, that word is associated with zoology, so that class has a higher probability. If the query changes to 'download python', then the most likely class changes.

- Update the likelihood of the class given new information.
- Prior probability (or knowledge that the label is one of the classes): Pr(y=Entertainment), Pr(y=Computer Science), Pr(y=Zoology).
- Posterior probability: Pr(y =Entertainment | x = "python") then is lower, because the class Computer science is higher
- Posterior probability = $\frac{Prior\ Probability * Likelihood}{Evidence} = P(y|X) = \frac{P(y)*P(X|y)}{P(X)}$ 3



- Pr(y=CS|“Python”) = $\frac{\text{Pr(y=CS) x Pr(“Python” | y=CS)}}{\text{Pr(“Python”)}}$
- Pr(y=Zoology|“Python”) = $\frac{\text{Pr(y=Zoology) x Pr(“Python” | y=Zoology)}}{\text{Pr(“Python”)}}$
- Pr(y=CS | “Python”) > Pr(y=Zoology | “Python”) $\Rightarrow$ y = CS

- In the algorithm, we just care for the maximum probability, and that is independent of the probability of the query (Pr of 'Python'), so in practical terms we end up with with the numeratior
- **Naive assumption: Given the class label, features are assumed to be independent of each other. This means that we calculate the bayes probability for each feature annd multiply.**

$$y^* = \underset{y}{\text{argmax}}\ \Pr(y \mid X) = \underset{y}{\text{argmax}}\ \Pr(y) \times \prod_{i=1}^{n} \Pr(x_i \mid y)$$

Query: "Python download"

$$y^* = \underset{y}{\text{argmax}}\ \Pr(y) \times \Pr(\text{"Python"} \mid y) \times \Pr(\text{"download"} \mid y)$$

Parameters

- Prior probabilities: Pr(y) for all y in Y. Probability of each class.
- Likelihood: Pr(xi | y) for all features xi and labels y in Y. Combination of every feature in each class.
- If there are 3 classes (|Y| = 3) and 100 features in X, how many parameters does naïve Bayes models have? 603.

Pr(x_i | y) for every binary feature x_i in X and y in Y. Specifically, for a particular feature x_1, the parameters are Pr(x_1 = 1 | y) and Pr(x_1 = 0 | y) for every y. So if |X| = 100 binary features and |Y| = 3, there are (2 x 100) x 3 = 600 such features.

- Learning parameters:
  - Prior probabilities: Pr(y) for all y in Y.
    - Count the number of instances n in each class of total N instances → **Pr(y) = n/N**
  - Likelihood: Pr(xi | y) for all features xi and labels y in Y.
    - Count how many times feature xi appears in instances labeled as class y.

        - If there are p instances of class y, and xi appears in k of those,Pr(xi |y)=k/p

Smoothing

What happens if Pr(xi | y) = 0. Meaning a feature doesn't appear in a given class. This is a problem because we are multiplying these probabilities. ⟶ Smooth the parameters.

- Laplace smoothing or Additive smoothing: Add a dummy count.
    - This doesn't change the overall result because it's just one in a larget set.
    - Pr(xi | y) = (k+1) / (p+n); where n is number of features

Conclusions

- Naïve Bayes is a probabilistic model.
- Naïve, because it assumes features are independent of each other given the class label. It doesn't give context or associate words. The probability of White followed by House is the same as White cat → All features are independent.
- For text classification problems, naïve Bayes models typically provide very strong baselines
  Simple model, easy to learn parameters

Naive Bayes Variations

Multinomial Naive Bayes

- Data follows a multinomial distribution
- Each future is independent of each other but can also appear multiple times, so counts become an important feature value (word occurrence counts, TF-IDF weighting, ….).

Bernoulli Naive Bayes (when freq. count is not important)

- Data follows a multivariate Bernoulli distribution.

- Each future is binary (present/absent).

Support vector machines

- Find a decision boundary that maximises separation between classes.
- Tend to be the most accurate classifiers, especially in high-dimensional data.
- Strong theoretical foundation
- Handles only numeric features.
  - Convert categorical features to numeric features.
  - Normalization is necesarry [0:1].
  - Hyperplane hard to interpret.

Linear boundary

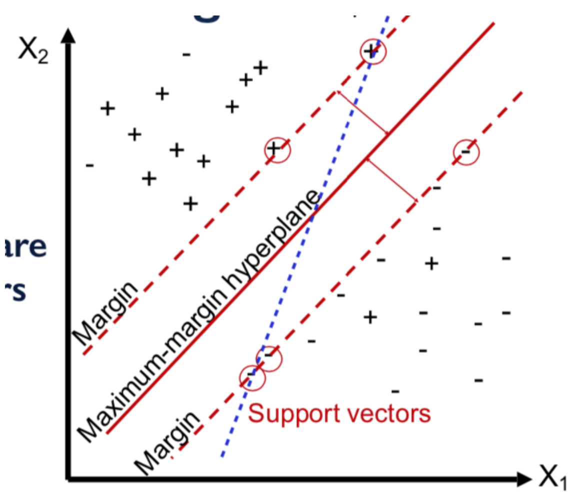
There are many methods to find the best linear boundary (Linear least squares, linear discriminative analysis, etc). SVM is one of them.

- By setting an area/band boundary, any small change in the data, would still be correctly classified and there is no need to change the boundary. It's more noise resilient.

- So in this context, SVM are maximum margin classifiers.
- SVMs are linear classifiers that find a hyperplane to separate two classes of data: positive and negative.

For **LINEAR** classification, SVM:

- Given training data (x1, y1), (x2, y2), … ; where xi = (x1,x2, …,xn) is instance vector and yi is one of {-1, +1},



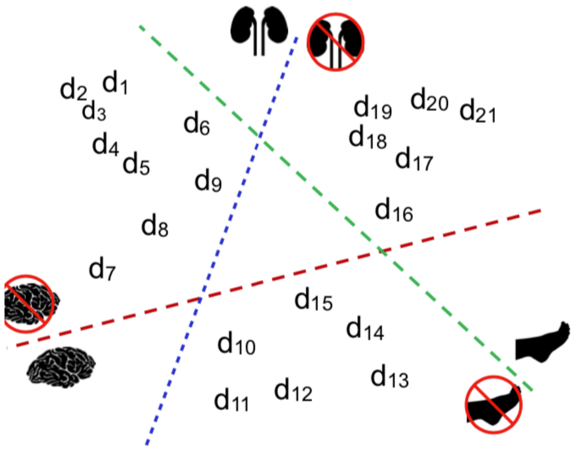
For **MULTI-CLASS** classification **One vs One**:

- SVM finds a linear function w (weight vector) so that f(xi) = < w . xi > + b if if f(xi) ≥ 0,yi = +1;else yi = -1

For **MULTI-CLASS** classification **One vs Rest**:

- A different classifier (binary classifiers) for each class.
- If there are n classes, there are n classifiers.

3 classes: Neurology, Podiatry, Nephrology

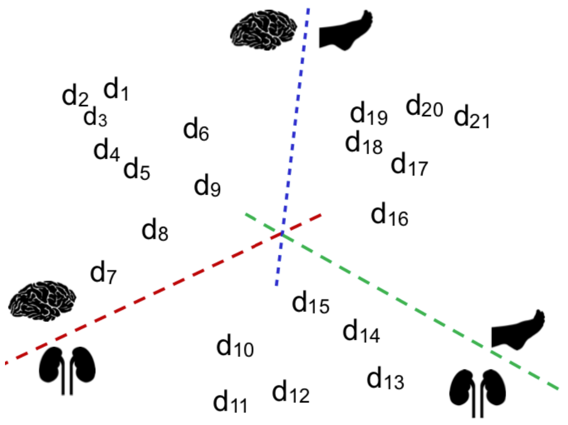


- A different classifier for each pair of classes.
- If there are n classes, there are C(n,2) classifiers.

$C_{n,x} = \binom{n}{x} = \frac{n!}{x!n-x!}$

- By looking at the resulting spaces, the majority class prevails.

3 classes: Neurology, Podiatry, Nephrology



SVM parameters

Parameter C

- Regularization: How much importance should you give individual data points as compared to better generalized model.
- Larger values of c = less regularization
  - Fit training data as well as possible, every data point important.
- Smaller values of c = more regularization.

    - More tolerant to errors on individual data points

Other params

- Linear kernels usually work best for text, but there are rbf, polynomial.
- multi_class: ovr (one-vs-rest).
- class_weight: Different classes can get different weights