07 Text Classification

Classification

Input: a document d (一般用 vector of features 表示 document)

Output: a fixed set of classes $C = \{c_1, c_2, \cdots, c_k\}$ (categorical, not continuous or ordinal)

Text Classification Tasks

Topic Classification

可以用于图书馆分类, 可以用于信息检索

Input:

- Unigram bag of words (BOW), with stop-words removed (for document)
- Longer n-grams (bigrams, trigrams) for phrases (for document)

Output class:

A topic (maybe select a topic from a topic set)

Examples of corpus:

- 新闻稿
- Tweets with hashtags

Sentiment Analysis

可以用于用户态度分析, 可以用于很多商业分析场景

Input:

- n-grams (for document)
- Polarity lexicons (需要预先准备一个极性词汇表, 记录各个词是有正面意义还是负面意义)

Output class:

Positive/Negative/(Neutral)

Examples of corpus:

- Polarity movie review dataset (in NLTK)
- SEMEVAL Twitter polarity datasets

Authorship Attribution 作者归属

预测一个匿名文本的作者是谁. 需要预先有一个作者class set, 结果只能从这些作者当中选择一个, 不能凭空给出一个新人. 可以用于forensic linguistics (司法语言学), plagiarism detection (抄袭检测).

Input:

- Frequency of function words (for document)
- Character n-grams (for document)
- Discourse structure (for document)

Output class:

An Author (每个作者成为一个class)

Examples of corpus:

- Project Gutenberg corpus (see NLTK sample)

Native-Language Identification

预测本文的作者的母语是什么. 可以用于forensic linguistics (司法语言学), educational applications.

Input:

- Word N-grams (for document)
- Syntactic patterns (POS, parse trees) (for document) 句法模式
- Phonological features (for document) 音韵特征

Output class:

First language of author (e.g.Chinese) (每个语言成为一个 class)

Examples of corpus:

- TOEFL/IELTS essay corpora

Automatic Fact-Checking

自动事实查核. 可用于假新闻 (fake news) 检测.

Input:

- N-grams (for document)
- Non-text metadata (for document)
- ….. 更多选项待开发

Output class:

True/False/Not Sure

Examples of corpus:

- Emergent, LIAR: political statements
- FEVER

Classification Algorithm

注意: well-annotated, plentiful datasets and appropriate features often more important than the specific algorithm used

Steps To Build a Text Classifier

1. Identify your task
2. Collect a corpus
3. Carry out annotation 进行注释
4. Select features (for documents)
5. Choose a machine learning classification algorithm
6. Tune hyper-parameters, train models, using development data to verify
7. If need, repeat above steps
8. Train a final model (如果一组超参在 development set 上令人满意了, 使用这组超参的 model 就可以用来做我们的 final model
9. Evaluate final model on held-out test data (用final model 来算出最终的 held-out test set performance)

Machine Learning Classification Algorithm

- Bias vs. Variance
- Feature independence
- Feature scaling
- Complexity
- Speed

Naive Bayes

NB 的基本思想是找到the class with highest probability under bales law.

注意: NB 算法天然地支持多分类问题

重要前提

无脑 assume features are independent, 因为只有independent 了才能使下方的 prob 计算公式成立. 但是有时候事实往往并不是如此, 很多时候 features 之间是 dependent 的, 如果 dependent 得厉害, 那么 NB 给出的结果就会很不对很离谱.

Prob 计算公式

这个公式计算了 given 一个 new data (一组 new features), 这个 new data 属于某个 class 的概率.

$$p(c_n|f_1 \ldots f_m) = \prod_{i=1}^{m} p(f_i|c_n)p(c_n)$$ where c_n is one of classes, $f_1 \ldots f_m$ are m features.

注意这个公式理论上只有在 features 满足 independent 前提的情况下才有效. 如果在 features dependent 的情况下强行套这个公式, 出来的结果其实是不对的, 很离谱的. 最后的 performance 也会很离谱.

优点:

- Fast to train (实际上不用train, 直接在 train set data套 prob 公式)
- Robust
- Low-variance (用 train set 的不同 subset 去 train model, train 出来的这些 model 对于一个 given data point 给出的结果不会相差很大, 这就是 low-variance)
- 当 independent 前提得到满足时, NB 算法是最好的选择

缺点:

- 当features 非常不 independent 时, NB 给出的结果就是在瞎搞
- 实际生活中, independent 前提很少能满足, 所以 NB 的 accuracy 常常不高.
- Smoothing required for unseen class-feature combinations. 不 smoothing 的话 $p(f_i|c_n)$ 就会=0, 乘一下之后最后probability 结果就必定是0了.也就是说,不 smoothing 的话, 只要一个 new data包含未出现过的class-feature combination, 这个新数据的预测结果就一定会死0.为了避免这种情况的发生,我们需要 smoothing.

Logistic Regression

LR 的基本思想是: 计算 result=1 的概率, 如果这个概率大于某个阈值, 那么判定为 1. 否则判定为 0

注意: LR 只天然地支持二分类

Prob 计算公式

$$p(class = 1 | f_1 \ldots f_m) = expit\left(w_0 + \sum_{i=1}^{m} w_i f_i\right)$$

Each feature has a weight, the weight value need to be trained

优点:

- 不像 NB 那样要求 feature independent. 即使 features 强烈 dependent LR 也不带怕的.
- Low-bias

缺点:

- Slow to train
- Some feature scaling issues
- Require a large train set to work well
- 常有 overfitting problem, 所以还需要 choose regularisation

Support Vector Machine

SVM基本思想是, 在dataset 空间中, 找一个超平面把两边 data 分开. 并且这个超平面跟两边的 data 离得越远越好 (margin 越大越好). 有些情况下, dataset实在无法用一个超平米分开, 那么 allow some misclassification.

注意: SVM只天然地支持二分类

优点:

- 基本款 SVM 是一个fast and accurate linear classifier (只能用平面分割dataset). 如果使用 kernel 就可以成为non-linearity classifier (可以用曲面分割dataset)
- 如果 feature 数量很多, SVM 也不怕. data 的features 再多, 都仍旧可以看做是分布在一个空间里.

缺点:

- 不支持多分类哦
- 需要feature scaling, 不然的话有些数字很大的 feature 会产生决定性的影响
- Deals poorly with unbalanced class. 如果train set 里面两个 class 的数量不均衡, 一个超多一个很少, SVM 就很难找到一个好的超平面去分割, 最后的 performance 也很可能不会好
- Uninterpretable

KNN

KNN 基本思想: 把 new data 放到 train set 空间中. 看看离这个 new data 距离最近的 k 个邻居中的majority class 是什么. 如果我的邻居大部分都说 1, 那我也跟风说 1, 如果邻居们大部分说 0, 那我也跟风说 0.

注意: 此处所说的 “距离” 的定义可以变动, 比如说有些情况下适合采用 Euclidean distance, 有些情况下适合采用 Cosine distance.

注意: KNN 天然地支持多分类问题

优点:

- 不需要 train, 直接放进去众数投票.
- optimal with infinite data.

缺点:

- 必须预先选定超参 k
- Deals poorly with unbalanced class. 如果train set 里面两个 class 的数量不均衡, 一个超多一个很少, 那么new data 会倾向于被分到那个多数的 class 去.
- Slow, 因为每次放进一个 new data, 需要把整个 dataset 都算一遍, 才能知道哪k个是最近的.
- Features must be selected carefully, 如果加入了一些不合适的 feature 可能会误导投票.

Decision Tree

每个 node 要导流时, 都只能以某一个 feature 的某一个值作为判定标准. 到了 leaf 才给出final class decision.

需要预先选定 tree-depth 等超参. 用greedy maximization of mutual information来 train, 决定 tree 需要多少 node 一个每个 node 用那个feature 什么值来导流.

注意: Decision tree 天然地支持多分类问题

优点:

- In theory, very interpretable.
- Fast to build and test
- Feature representation/scaling irrelevant
- Good for small feature sets
- Handles non-linearly-separable problems

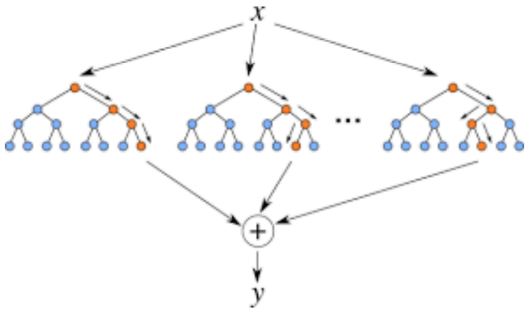
缺点:

- In practice, often not that interpretable
- Highly redundant sub-trees
- large-feature sets 不适用, 太慢

Random Forest

用 train set 的多个 subset 各自 train a decision tree. 然后把一个 new data 放到这些 decision tree 里面, 每个 tree 都会给出一个判定结果(一个 class). 然后我们再用某种方法(比如众数投票)把这些结果 ensemble 起来, 形成一个最终结果.

注意: Random forest 天然地支持多分类问题



优点:

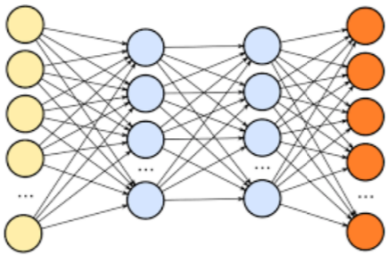
- Usually more accurate and more robust than decision trees
- a great classifier for small- to moderate- sized feature sets
- training easily parallelised

缺点:

- 和 decision tree 一样, large-feature sets 不适用, 太慢

Neural Network

Input layer (features), output layer (class probabilities),
and one or more hidden layers. 每条线都代表着一个不同的 weight, 每个 hidden layer 都有一个激活函数.



优点:

- Extremely powerful, state-of-the-art accuracy on many tasks in natural language processing and vision

缺点:

- Not an off-the-shelf classifier, very difficult to choose good hyper-parameters (层数, 每层节点数)
- slow to train. weight 是由训练得到的
- prone to overfitting

调参

- 用 development set 调参
- 可以用 k-fold cross-validation 调参
- But many hyper-parameters are for Regularization (penalize model complexity to prevent overfitting)
- For multiple hyper-parameters, use grid search

Evaluation

Accuracy

- Accuracy = 判定对的(TP+TN)/所有

Precision

- 给出的 Positive 判定中, 有多少个是真的 Positive?

Precision = P正确判定为P的(TP)/判定了多少个 P (TP+FP)

Recall

- 有多少条 Positive 数据被找出出来了(被成功地判定成 Positive) ?

Recall = P正确判定为P的(TP)/真正的 P 有多少个(TP+FN)

F1-Score

Harmonic mean of precision and recall

$$F1 = \frac{2 * precision * recall}{(precision + recall)}$$

- Macro average: Average F1-score across classes
- Micro average: Calculate F1-score using sum of counts