Một lần nữa,
γ
có thể được tinh chỉnh thông qua kiểm định chéo để đạt được hiệu suất tốt nhất. Một số hàm nhân (kernel) thông dụng khác bao gồm hàm nhân đa thức và hàm nhân sigmoid:

$$K_{\text{polynomial}}(x^{(i)}, x^{(j)}) = (x^{(i)} \cdot x^{(j)} + r)^d$$

$$K_{\text{sigmoid}}(x^{(i)}, x^{(j)}) = \tanh(\gamma(x^{(i)} \cdot x^{(j)}) + r)$$

Trong trường hợp không có kiến thức ưu tiên của chuyên gia về phân phối, RBF thường được ưa chuộng hơn trong thực tế, vì có nhiều tham số hơn (bậc đa thức d cho hàm nhân đa thức và hàm nhân sigmoid được xác định dựa trên kinh nghiệm) để điều chỉnh cho hàm nhân đa thức và hàm nhân sigmoid có thể hoạt động ngang hàng với RBF chỉ dưới một số điều kiện nhất định. Do đó, cuộc tranh luận chủ yếu là giữa hàm nhân tuyến tính và RBF.

Lựa chọn giữa kernel tuyến tính và RBF

Quy tắc chung, tất nhiên, là khả năng phân tách tuyến tính. Tuy nhiên, đây là điều hầu như rất khó để xác định, trừ khi bạn có kiến thức nền tảng đầy đủ hoặc các đặc trưng có số chiều thấp (từ 1 đến 3).

Kiến thức nền tảng, bao gồm dữ liệu văn bản, thường có thể phân tách tuyến tính, dữ liệu từ hàm xor thì không, và chúng ta sẽ xem xét ba kịch bản sau đây nơi mà kernel tuyến tính được ưa chuộng hơn RBF:

Trường hợp 1: cả số lượng đặc trưng và thể hiện đều lớn (nhiều hơn 10^4 hoặc 10^5). Khi mà không gian đặc trưng có chiều cao đủ lớn, thêm đặc trưng từ sự biến đổi của RBF sẽ không cung cấp bất kỳ cải thiện hiệu suất nào, nhưng sẽ tăng chi phí tính toán. Một số ví dụ từ Kho Dữ liệu Học Máy UCI của loại này:

- **Dữ liệu Reputation URL Set:** https://archive.ics.uci.edu/ml/datasets /URL+ Reputation (số lượng trường hợp: 2396130, số tính năng: 3231961) để phát hiện URL độc hại dựa trên thông tin từ vựng và máy chủ của chúng
- **Bộ dữ liệu trò chơi điện tử nhiều lượt xem trên YouTube:** https://archive.ics.uci.edu/ml/datasets/YouTube+Multiview+Video+G (số lượng phiên bản: 120000, số lượng tính năng: 1000000) cho chủ đề phân loại

Trường hợp 2: số lượng đặc điểm đáng chú ý lớn so với số lượng mẫu huấn luyện. Ngoài các lý do được nêu trong Kịch bản 1, hạt nhân RBF có xu hướng dễ bị quá mức khớp hơn đáng kể. Một tình huống như vậy xảy ra trong, ví dụ:

- **Bộ dữ liệu Dorothea:** https://archive.ics.uci.edu/ml/datasets/Dorothea (số trường hợp: 1950, số đặc điểm: 100000) đối với thuốc khám phá phân loại các hợp chất hóa học là hoạt động hoặc không hoạt động bằng cách đặc điểm cấu trúc phân tử
- **Bộ dữ liệu Arcene:** https://archive.ics.uci.edu/ml/datasets/Arcene (số lượng trường hợp: 900, số lượng tính năng: 10000) bộ dữ liệu khối phổ để phát hiện ung thư

Trường hợp 3: số lượng các thể hiện là đáng kể lớn hơn so với số lượng các đặc trưng. Đối với một tập dữ liệu có kích thước thấp, hạt nhân RBF nói chung sẽ tăng cường hiệu suất bằng cách ánh xạ nó lên một không gian chiều cao hơn. Tuy nhiên, do độ phức tạp của quá trình huấn luyện, nó thường không còn hiệu quả trên một tập huấn luyện với hơn 10^6 hoặc 10^7 mẫu. Một số tập dữ liệu điển hình bao gồm:

- **Dữ liệu nhận dạng hoạt động không đồng nhất:** https://archive.ics.uci.edu/ml/datasets/Heterogeneity+Activity+Reco số trường hợp: 43930257, số tính năng: 16) cho hoạt động của con người sự công nhận
- **Tập dữ liệu HIGGS:** https://archive.ics.uci.edu/ml/datasets/HIGGS (số lượng phiên bản: 11000000, số lượng tính năng: 28) cho phân biệt giữa quá trình tín hiệu tạo ra boson Higgs hay một quá trình nền

Ngoài ba trường hợp trước đó, RBF hầu như là lựa chọn đầu tiên. Các quy tắc để lựa chọn giữa hạt nhân tuyến tính và RBF có thể được tóm tắt như sau:

Case	Linear	RBF
Expert prior knowledge	If linearly separable	If nonlinearly separable
Visualizable data of 1 to 3 dimension	If linearly separable	If nonlinearly separable
Both numbers of features and instances are large	First choice	
Features ≫ Instances	First choice	
Instances ≫ Features	First choice	
Others		First choice

Phân loại chủ đề tin tức với support vector machine

Đã đến lúc chúng ta xây dựng bộ phân loại chủ đề tin tức dựa trên SVM, tân tiến nhất với tất cả những gì chúng ta vừa học được.
Tải và làm sạch bộ dữ liệu tin tức với toàn bộ 20 nhóm:

```
>>> categories = None
>>> data_train = fetch_20newsgroups(subset='train',
                                    categories=categories, random_state=
>>> data_test = fetch_20newsgroups(subset='test',
                                    categories=categories, random_state=
>>> cleaned_train = clean_text(data_train.data)
>>> label_train = data_train.target
>>> cleaned_test = clean_text(data_test.data)
>>> label_test = data_test.target
>>> term_docs_train =
                  tfidf_vectorizer.fit_transform(cleaned_train)
>>> term_docs_test = tfidf_vectorizer.transform(cleaned_test)
```

Hãy nhớ rằng nhân tuyến tính rất tốt để phân loại dữ liệu văn bản; chúng ta tiếp tục thiết lập giá trị nhân là linear cho tham số kernel trong mô hình SVC và chúng ta chỉ cần điều chỉnh hình phạt c thông qua kiểm định chéo:

```
>>> svc_libsvm = SVC(kernel='linear')
```

Phương pháp mà chúng ta đã thực hiện kiểm định chéo cho đến nay là để rõ ràng phân chia dữ liệu thành các fold và lặp lại viết vòng lặp for để liên tục kiểm tra từng tham số. Chúng ta sẽ giới thiệu một cách tiếp cận duyên dáng hơn bằng cách sử dụng công cụ GridSearchCV từ scikit-learn. GridSearchCV xử lý toàn bộ quá trình một cách ngầm định, bao gồm việc phân chia dữ liệu, tạo fold, huấn luyện chéo và kiểm định, và cuối cùng là tìm kiếm triệt để qua bộ tham số tốt nhất. Những gì còn lại cho chúng ta là chỉ cần chỉ định tham số (các tham số) để điều chỉnh và các giá trị để khám phá cho từng tham số cá nhân:

```
>>> parameters = {'C': (0.1, 1, 10, 100)}
```

```
>>> from sklearn.model_selection import GridSearchCV
>>> grid_search = GridSearchCV(svc_libsvm, parameters,
                                           n_jobs=-1, cv=3)
```

Mô hình GridSearchCV mà chúng ta vừa khởi tạo sẽ thực hiện kiểm định chéo 3 lần (cv=3) và sẽ chạy đồng thời trên tất cả các nhân có sẵn (n_jobs=-1). Sau đó, chúng ta tiến hành điều chỉnh tham số mô hình bằng cách đơn giản áp dụng phương thức fit, và ghi lại thời gian chạy:

```
>>> import timeit
>>> start_time = timeit.default_timer()
>>> grid_search.fit(term_docs_train, label_train)
>>> print("--- %0.3fs seconds ---" % (
                              timeit.default_timer() - start_tim
--- 189.506s seconds ---
```

Chúng ta có thể thu được bộ tham số tối ưu (C tối ưu trong trường hợp này) bằng cách như sau:

```
>>> grid_search.best_params_
{'C': 10}
```

Và hiệu suất trung bình gấp 3 lần tốt nhất theo bộ thông số tối ưu:

```
>>> grid_search.best_score_
0.8665370337634789
```

Sau đó, chúng tôi truy xuất mô hình SVM với tham số tối ưu và áp dụng nó cho bộ thử nghiệm chưa xác định:

```
>>> svc_libsvm_best = grid_search.best_estimator_
>>> accuracy = svc_libsvm_best.score(term_docs_test, label_test
>>> print('The accuracy on testing set is:
                                {0:.1f}%'.format(accuracy*10(
The accuracy on testing set is: 76.2%
```

Cần lưu ý rằng chúng tôi điều chỉnh mô hình dựa trên bộ dữ liệu huấn luyện gốc, được chia thành các fold để huấn luyện chéo và đánh giá, và chúng tôi chọn mô hình tối ưu để áp dụng cho bộ dữ liệu kiểm tra gốc. Chúng tôi xem xét hiệu suất phân loại theo cách này để đo lường mức độ tổng quát hóa của mô hình là như thế nào, mục tiêu là để đưa ra dự đoán chính xác trên một bộ dữ liệu hoàn toàn mới. Một độ chính xác phân loại là 76.2% đã được đạt được với mô hình SVC đầu tiên của chúng tôi. Liệu một mô hình SVM phân loại khác, LinearSVC, từ scikit-learn sẽ thể hiện như thế nào? LinearSVC tương tự như SVC với kernel tuyến tính, nhưng nó được thực hiện dựa trên thư viện liblinear thay vì libsvm. Chúng tôi lặp lại quá trình trước đó giống hệt cho LinearSVC:

```
>>> from sklearn.svm import LinearSVC
>>> svc_linear = LinearSVC()
>>> grid_search = GridSearchCV(svc_linear, parameters,
                                                n_jobs=-1, cv=3
>>> start_time = timeit.default_timer()
>>> grid_search.fit(term_docs_train, label_train)
>>> print("--- %0.3fs seconds ---" %
                           (timeit.default_timer() - start_tim
--- 16.743s seconds ---
>>> grid_search.best_params_
{'C': 1}
>>> grid_search.best_score_
0.8707795651405339
>>> svc_linear_best = grid_search.best_estimator_
>>> accuracy = svc_linear_best.score(term_docs_test, label_test
>>> print('The accuracy on testing set is:
                                    {0:.1f}%'.format(accuracy*1
The accuracy on testing set is: 77.9%
```

Mô hình LinearSVC vượt trội hơn mô hình SVC và đặc biệt huấn luyện nhanh hơn đến 10 lần. Điều này là do thư viện liblinear có khả năng mở rộng cao được thiết kế cho các bộ dữ liệu lớn trong khi thư viện libsvm với độ phức tạp tính toán hơn bậc tứ không thể mở rộng tốt với hơn 105 trường hợp huấn luyện.

Chúng tôi cũng có thể điều chỉnh bộ trích xuất đặc trưng, mô hình TfidfVectorizer, để cải thiện thêm hiệu suất. Bước trích xuất đặc trưng và phân loại như hai bước liên tiếp nên được kiểm định chéo cùng nhau. Chúng tôi sử dụng API pipeline từ scikit-learn để hỗ trợ điều này.

Bộ trích xuất đặc trưng Tfidf và phân loại SVM tuyến tính đầu tiên được lắp ráp trong pipeline:

```
>>> from sklearn.pipeline import Pipeline
>>> pipeline = Pipeline([
...     ('tfidf', TfidfVectorizer(stop_words='english')),
...     ('svc', LinearSVC()),
... ])
```

Các tham số của cả hai bước cần được điều chỉnh được định nghĩa như sau, với tên bước trong pipeline được nối với tên tham số bởi một dấu gạch dưới _ làm khóa, và một bộ các tùy chọn tương ứng làm giá trị:

```
>>> parameters_pipeline = {
...     'tfidf__max_df': (0.25, 0.5),
...     'tfidf__max_features': (40000, 50000),
...     'tfidf__sublinear_tf': (True, False),
...     'tfidf__smooth_idf': (True, False),
...     'svc__C': (0.1, 1, 10, 100),
... }
```

Ngoài việc điều chỉnh hình phạt C cho bộ phân loại SVM, chúng tôi cũng tinh chỉnh trình trích xuất đặc trưng tfidf theo các tiêu chí:

- max_df: Tần suất xuất hiện tối đa của một từ trong tài liệu, nhằm tránh những từ thường gặp trong các tài liệu
- max_features: Số lượng đặc trưng hàng đầu để xem xét; cho đến nay chúng tôi chỉ sử dụng 8000 đặc trưng cho mục đích thử nghiệm
- sublinear_tf: Có sử dụng hàm logarit để chia tỷ lệ tần suất từ hay không
- smooth_idf: Thêm một số ban đầu vào tần suất tài liệu hay không, tương tự như việc làm mịn cho tần suất từ
- Mô hình tìm kiếm lưới tìm kiếm bộ tham số tối ưu trong suốt toàn bộ quy trình:

```
>>> grid_search = GridSearchCV(pipeline, parameters_pipeline,
                               n_jobs=-1, cv=3)
>>> start_time = timeit.default_timer()
>>> grid_search.fit(cleaned_train, label_train)
>>> print("--- %0.3fs seconds ---" %
                    (timeit.default_timer() - start_time))
--- 278.461s seconds ---
>>> grid_search.best_params_
{'tfidf__max_df': 0.5, 'tfidf__smooth_idf': False,
 'tfidf__max_features': 40000, 'svc__C': 1,
 'tfidf__sublinear_tf': True}
>>> grid_search.best_score_
0.888368393141240 94
>>> pipeline_best = grid_search.best_estimator_
```

Và cuối cùng được áp dụng cho bộ thử nghiệm:

```
>>> accuracy = pipeline_best.score(cleaned_test, label_test)
```

```
>>> print('The accuracy on testing set is: {0:.1f}%'.format(acc
The accuracy on testing set is: 80.6%
```

Bộ cài đặt (max_df: 0.5, smooth_idf: False, max_features: 40000, sublinear_tf: True, C: 1) cho phép đạt được độ chính xác phân loại tốt nhất, 80.6% trên toàn bộ 20 nhóm dữ liệu tin tức.

Thêm ví dụ - phân loại trạng thái thai nhi trên chụp tim mạch bằng SVM

Sau một ứng dụng thành công của SVM với nhân tuyến tính, chúng tôi xem xét thêm một ví dụ nữa nơi SVM với nhân RBF phù hợp cho nó.

Chúng tôi sẽ xây dựng một phân loại giúp các bác sĩ sản khoa phân loại các đồ thị tim thai (CTGs) thành một trong ba trạng thái sau: bình thường, nghi ngờ và bệnh lý. Bộ dữ liệu về đồ thị tim thai mà chúng tôi sử dụng đến từ http://archive.ics.uci.edu/ml/datasets/Cardiotocography và có thể được tải trực tiếp từ UCI Machine Learning Repository và có thể được tải trực tiếp qua http://archive.ics.uci.edu/ml/machine-learning-databases/00193/CTG.xls dưới dạng tệp Excel .xls. Bộ dữ liệu bao gồm các phép đo về nhịp tim thai và co bóp tử cung như là các đặc trưng và mã trạng thái thai nhi (1= bình thường, 2= nghi ngờ, 3= bệnh lý) như là nhãn. Tổng cộng có 2126 mẫu với 23 đặc trưng. Dựa trên số lượng mẫu và đặc trưng (2126 không quá lớn so với 23), nhân RBF là lựa chọn hàng đầu.

Ở đây chúng tôi làm việc với tệp .xls Excel sử dụng pandas (http://pandas.pydata.org/), đây là một thư viện phân tích dữ liệu mạnh mẽ. Nó có thể được cài đặt dễ dàng với lệnh pip install pandas trong Terminal. Có thể yêu cầu cài đặt thêm gói xlrd, mà mô-đun Excel của pandas dựa vào.

Chúng tôi đầu tiên đọc dữ liệu được định vị trong bảng có tên 'Raw Data':

```
>>> import pandas as pd
>>> df = pd.read_excel('CTG.xls', "Raw Data")
```

Sau đó lấy 2126 mẫu dữ liệu này, gán bộ tính năng (từ cột D đến AL trong bảng tính) và bộ nhãn (cột AN) tương ứng:

```
>>> X = df.ix[1:2126, 3:-2].values
>>> Y = df.ix[1:2126, -1].values
```

Đừng quên kiểm tra tỷ lệ các lớp:

```
>>> Counter(Y)
Counter({1.0: 1655, 2.0: 295, 3.0: 176})
```

Chúng tôi dành riêng 20% dữ liệu gốc cho việc kiểm tra cuối cùng:

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                        test_size=0.2, random_state=
```

Bây giờ chúng ta điều chỉnh mô hình SVM dựa trên RBF về mặt hình phạt C và hệ số nhân của hạt nhân:

```
γ

:

>>> svc = SVC(kernel='rbf')
>>> parameters = {'C': (100, 1e3, 1e4, 1e5),
...               'gamma': (1e-08, 1e-7, 1e-6, 1e-5)}
>>> grid_search = GridSearchCV(svc, parameters, n_jobs=-1, cv=3
>>> start_time = timeit.default_timer()
>>> grid_search.fit(X_train, Y_train)
>>> print("--- %0.3fs seconds ---" %
                      (timeit.default_timer() - start_time))
--- 6.044s seconds ---
>>> grid_search.best_params_
{'C': 100000.0, 'gamma': 1e-07}
>>> grid_search.best_score_
0.942352941176
>>> svc_best = grid_search.best_estimator_
```

Và cuối cùng sử dụng mô hình tối ưu cho tập thử nghiệm:

```
>>> accuracy = svc_best.score(X_test, Y_test)
>>> print('The accuracy on testing set is:
                                {0:.1f}%'.format(accuracy*100))
The accuracy on testing set is: 96.5%
```

Đồng thời kiểm tra hiệu suất của từng lớp vì dữ liệu không hoàn toàn cân bằng:

```
>>> prediction = svc_best.predict(X_test)
```

```
>>> report = classification_report(Y_test, prediction)
>>> print(report)
             precision    recall  f1-score   support

        1.0       0.98      0.98      0.98       333
        2.0       0.89      0.91      0.90        64
        3.0       0.96      0.93      0.95        29
avg / total       0.96      0.96      0.96       426
```

Tóm tắt

Trong chương này, chúng tôi đã mở rộng kiến thức về trích xuất đặc trưng văn bản bằng cách giới thiệu một kỹ thuật tiên tiến được gọi là tần số-ngược tần số tài liệu (frequency-inverse document frequency). Sau đó, chúng tôi tiếp tục hành trình phân loại dữ liệu tin tức với máy vector hỗ trợ (support vector machine classifier), nơi chúng tôi đã nắm bắt được cơ chế của SVM, các kỹ thuật và triển khai SVM, cũng như các khái niệm quan trọng của phân loại học máy, bao gồm phân loại đa lớp (multiclass classification) và các chiến lược học sâu, cũng như các vấn đề cụ thể cho việc sử dụng SVM (ví dụ, lựa chọn giữa các hạt nhân và điều chỉnh tham số). Cuối cùng, chúng tôi đã chọn những gì phù hợp nhất với hai trường hợp thực tế, phân loại chủ đề tin tức và phân loại trạng thái thai nhi.

Chúng tôi đã học và áp dụng hai thuật toán phân loại cho đến nay, Bayes ngây thơ và SVM. Bayes ngây thơ là một thuật toán đơn giản. Đối với một tập dữ liệu có các đặc trưng độc lập, Bayes ngây thơ thường hoạt động tốt. SVM linh hoạt để thích nghi với tính tách biệt tuyến tính của dữ liệu. Nói chung, độ chính xác rất cao có thể đạt được bằng cách sử dụng SVM với hạt nhân và tham số phù hợp. Tuy nhiên, điều này có thể tốn kém về mặt tính toán cường độ cao và tiêu thụ bộ nhớ lớn. Khi so sánh SVM với hạt nhân tuyến tính và Bayes ngây thơ, chúng thường có hiệu suất tương đương. Trong thực tế, chúng ta có thể thử nghiệm cả hai và chọn lựa cái tốt hơn với tham số tối ưu.

Chương 5. Dự đoán Click-Through với Các Thuật toán Dựa trên Cây

Trong chương này và chương tiếp theo, chúng ta sẽ giải quyết một trong những vấn đề quan trọng nhất của máy học trong quảng cáo trực tuyến số, đó là dự đoán click-through—cụ thể là khi một người dùng và trang họ đang xem, khả năng họ click vào một quảng cáo là bao nhiêu. Chúng ta sẽ tập trung vào việc học các thuật toán dựa trên cây, bao gồm cây quyết định và rừng ngẫu nhiên, và sử dụng chúng để giải quyết vấn đề tỷ đô.

Chúng ta sẽ đi sâu vào các chủ đề được đề cập sau:

- Giới thiệu về click-through trong quảng cáo trực tuyến
- Hai loại đặc trưng, số và hạng mục
- Phân loại cây quyết định
- Cơ chế của cây quyết định
- Xây dựng cây quyết định
- Cài đặt cây quyết định
- Dự đoán click-through với cây quyết định
- Rừng ngẫu nhiên
- Cơ chế của rừng ngẫu nhiên
- Dự đoán click-through với rừng ngẫu nhiên
- Điều chỉnh mô hình rừng ngẫu nhiên

Tổng quan ngắn gọn về dự đoán tỷ lệ nhấp chuột quảng cáo

Quảng cáo trực tuyến dạng hiển thị là một ngành công nghiệp đa tỷ đô. Nó xuất hiện dưới nhiều định dạng khác nhau bao gồm các biểu ngữ quảng cáo bằng văn bản, hình ảnh, flash, và các định dạng phong phú khác bao gồm video. Quảng cáo của các doanh nghiệp xuất hiện trên nhiều loại trang web, thậm chí cả ứng dụng di động trên Internet, để tiếp cận người tiêu dùng một cách rộng rãi và đa dạng hơn.

Quảng cáo trực tuyến dạng hiển thị đã được coi là một trong những ví dụ tốt nhất về việc sử dụng học máy. Rõ ràng, nhà quảng cáo cũng như người tiêu dùng đều được hưởng lợi từ việc tối ưu hóa quảng cáo trực tuyến. Các nhà quảng cáo dựa vào học máy để dự đoán khả năng một người tiêu dùng cụ thể sẽ quan tâm đến sản phẩm của họ; còn người tiêu dùng thì dựa vào học máy để nhận được các quảng cáo phù hợp với sở thích và nhu cầu của họ. Các mô hình học máy có thể dự đoán sự quan tâm của người tiêu dùng dựa trên lịch sử duyệt web, hành vi mua sắm, và các yếu tố khác để đưa ra quảng cáo phù hợp nhất.

Một trong những chỉ số quan trọng nhất trong quảng cáo trực tuyến là tỷ lệ nhấp chuột (CTR), đây là tỷ lệ giữa số lần nhấp vào quảng cáo so với số lần hiển thị. CTR càng cao, chiến dịch quảng cáo trực tuyến càng thành công.

Dự đoán tỷ lệ nhấp chuột mang lại cả hứa hẹn và thách thức cho máy học. Máy học sẽ phân tích một lượng lớn dữ liệu nhị phân để dự đoán việc người dùng có nhấp vào quảng cáo hay không. Nó bao gồm nhiều tính năng khác nhau như:

- Nội dung quảng cáo và thông tin (danh mục, vị trí, định dạng, v.v.)
- Nội dung trang (thông tin trang, ngữ cảnh, tên miền, và v.v.)
- Thông tin người dùng (tuổi, giới tính, thu nhập, sở thích, lịch sử tìm kiếm, thiết bị sử dụng, v.v.)

Giả sử chúng ta, là một đại lý, đang điều hành quảng cáo thay mặt cho nhiều nhà quảng cáo và công việc của chúng ta là hiển thị quảng cáo đúng đắn cho đối tượng khán giả phù hợp. Với một bộ dữ liệu hiện có trong tay (đoạn nhỏ sau đây chỉ là một ví dụ, số lượng các đặc trưng dự đoán có thể dễ dàng lên tới hàng nghìn trong thực tế) được lấy từ hàng triệu bản ghi của các chiến dịch quảng cáo diễn ra trong tháng trước, chúng ta cần phát triển một mô hình phân loại để học hỏi và dự đoán kết quả đặt quảng cáo trong tương lai.

Ad category	Site category	Site domain	User age	User gender	User occupation	Interested in sports	Interested in tech	Click
Auto	News	cnn.com	25-34	M	Professional	True	True	1
Fashion	News	bbc.com	35-54	F	Professional	False	False	0
Auto	Edu	onlinestudy.com	17-24	F	Student	True	True	0
Food	Entertainment	movie.com	25-34	M	Clerk	True	False	1
Fashion	Sports	football.com	55+	M	Retired	True	False	0
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…

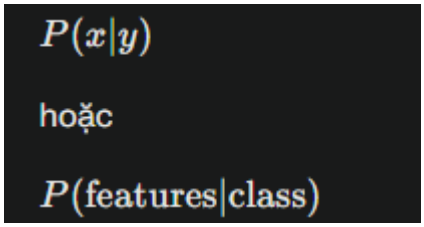
Food	News	abc.com	17-24	M	Student	True	True	**?**
Auto	Entertainment	movie.com	35-54	F	Professional	True	False	**?**

Bắt đầu với hai loại dữ liệu, số và phân loại.

Các đặc trưng phân loại (còn được gọi là đặc trưng định tính) đại diện cho các đặc tính, nhóm riêng biệt và có một số lượng lựa chọn đếm được. Các đặc trưng phân loại có thể hoặc không có thứ tự logic. Ví dụ, thu nhập hộ gia đình từ thấp, trung bình đến cao là một đặc trưng thứ tự, trong khi loại của một quảng cáo không phải là thứ tự. Các đặc trưng số (còn được gọi là định lượng), ngược lại, có ý nghĩa toán học như một phép đo và tất nhiên là có thứ tự. Ví dụ, tần suất chuyển động và biến thể tf-idf lần lượt là rời rạc (như số lần gia tốc mỗi giây, số lượng chuyển động thai nhi) và liên tục (như giá trị trung bình của biến động dài hạn) là các đặc trưng số.

Các đặc trưng phân loại cũng có thể mang giá trị số. Ví dụ, từ 1 đến 12 đại diện cho các tháng trong năm, và 1 và 0 chỉ nam và nữ. Tuy nhiên, những giá trị này không chứa ý nghĩa toán học.

Trong số hai thuật toán phân loại, naive Bayes và SVM, mà chúng ta đã học trước đây, phân loại naive Bayes hoạt động cho cả đặc trưng số và phân loại như là xác suất có điều kiện



được tính toán theo cùng một cách, trong khi SVM yêu cầu đặc trưng phải là số để tính toán biên độ.

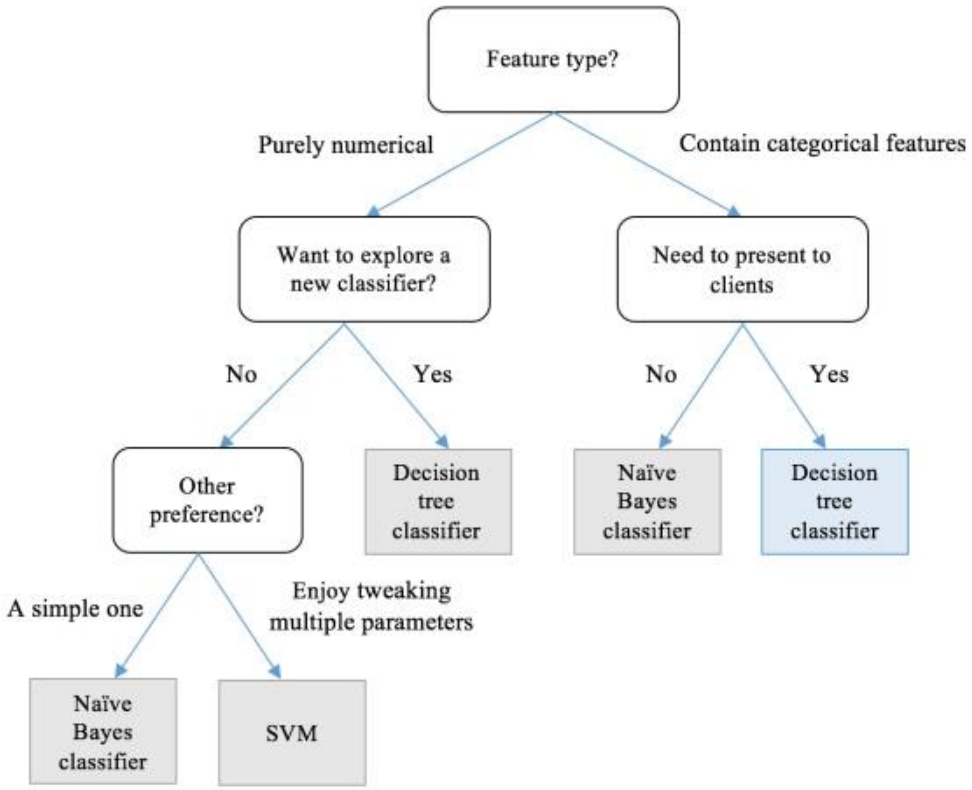
Bây giờ, nếu chúng ta nghĩ đến việc dự đoán click hoặc không click với Naive Bayes, và cố gắng giải thích mô hình cho khách hàng quảng cáo của chúng ta, khách hàng của chúng ta sẽ thấy khó hiểu về xác suất trước của từng thuộc tính và việc nhân chúng lại với nhau. Có phân loại nào dễ hiểu, giải thích cho khách hàng, và cũng có khả năng xử lý dữ liệu phân loại không?

Cây quyết định!

Bộ phân loại cây quyết định

Cây quyết định là một dạng đồ thị giống như cây, một sơ đồ tuần tự minh họa tất cả các phương án quyết định có thể và kết quả tương ứng. Bắt đầu từ gốc của cây, mỗi nút nội bộ đại diện cho một quyết định được đưa ra; mỗi nhánh của nút đại diện cho việc một lựa chọn có thể dẫn đến các nút tiếp theo; và cuối cùng, mỗi nút lá, tức là nút cuối cùng, đại diện cho một kết quả được đưa ra.

Ví dụ, chúng ta vừa mới đưa ra một vài quyết định đã đưa chúng ta đến hành động học cây quyết định để giải quyết vấn đề quảng cáo của mình:

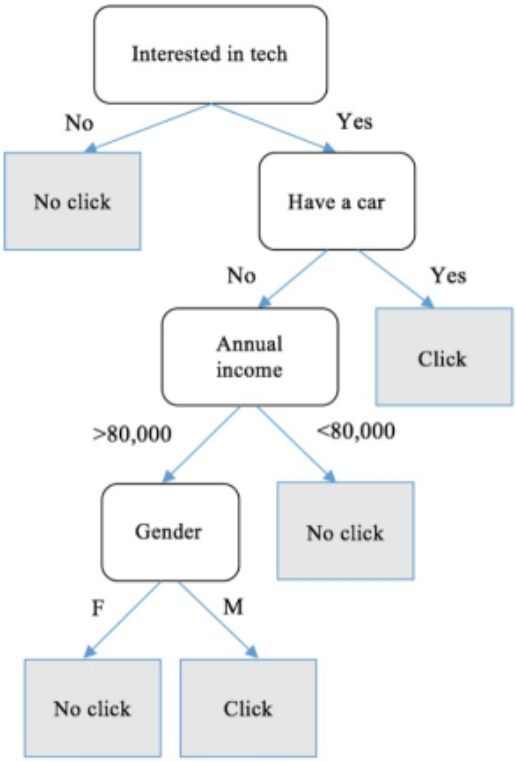


Bộ phân loại cây quyết định hoạt động dưới dạng một cây quyết định. Nó ánh xạ các quan sát vào các phân loại lớp (biểu tượng hóa như các nút lá), thông qua một chuỗi các bài kiểm tra (được biểu diễn như các nút nội bộ) dựa trên giá trị đặc trưng và các điều kiện tương ứng (được biểu diễn như các nhánh). Tại mỗi nút, một câu hỏi liên quan đến giá trị và đặc tính của một đặc trưng được đặt ra; dựa trên câu trả lời cho câu hỏi, các quan sát được chia thành các tập con. Các bài kiểm tra tuần tự được tiến hành cho đến khi kết luận về nhãn mục tiêu của các quan sát được đạt tới. Các con đường từ gốc đến lá cuối cùng biểu diễn quá trình ra quyết định, các quy tắc phân loại.

Hình ảnh sau đây mô tả một kịch bản đơn giản hóa, nơi chúng ta muốn dự đoán việc nhấp hoặc không nhấp vào quảng cáo xe tự lái, chúng ta đã tự tạo một bộ phân loại cây quyết định hoạt động cho một bộ dữ liệu có sẵn. Ví dụ, nếu người dùng quan tâm đến công nghệ và họ có xe hơi, họ sẽ có xu hướng nhấp vào quảng cáo; đối với một người nằm ngoài nhóm người này, nếu người đó là phụ nữ có thu nhập cao, thì họ sẽ không có khả năng nhấp vào quảng cáo. Sau đó, chúng ta sử dụng cây quyết định đã học để dự đoán hai đầu vào mới, kết quả tương ứng là nhấp và không nhấp.

User gender	Annual income	Have a car	Interested in tech	Click
M	200,000	True	True	1
F	5,000	False	False	0
F	100,000	True	True	1
M	10,000	True	False	0
M	80,000	False	False	0
...	...	...	...	...
...	...	...	...	...

M	120,000	True	True	?
F	70,000	False	True	?



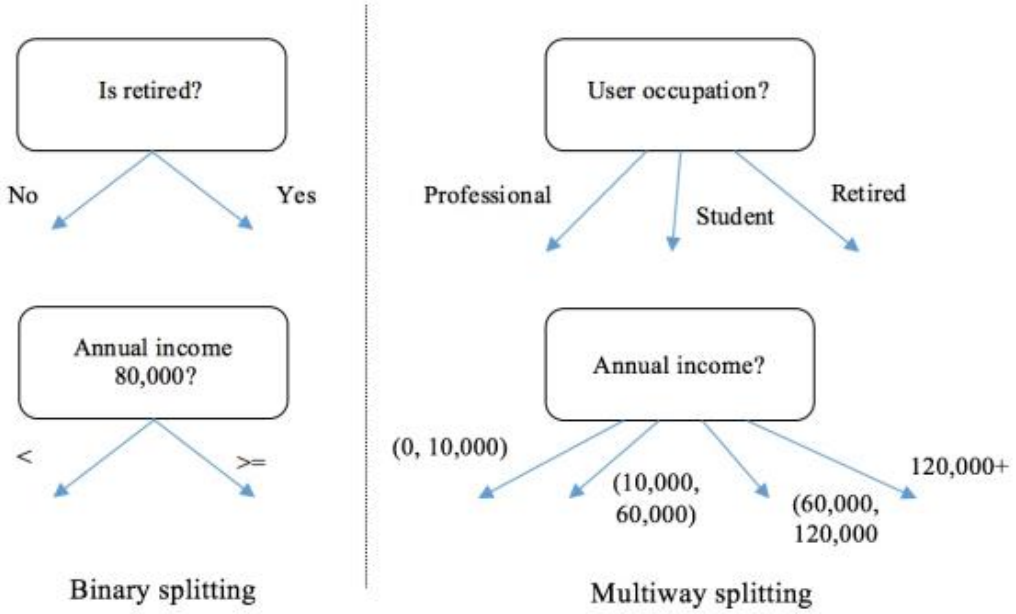
Sau khi một cây quyết định đã được xây dựng, việc phân loại một mẫu mới là đơn giản như chúng ta vừa thấy: bắt đầu từ nút gốc, áp dụng điều kiện kiểm tra và theo dõi nhánh tương ứng cho đến khi một nút lá được tiếp cận và nhãn lớp liên quan sẽ được gán cho mẫu mới.

Vậy làm thế nào chúng ta có thể xây dựng một cây quyết định phù hợp?

Xây dựng cây quyết định

Một cây quyết định được xây dựng bằng cách phân chia các mẫu huấn luyện thành các tập con liên tiếp. Quá trình phân chia được lặp lại theo cách đệ quy trên mỗi tập con. Đối với mỗi lần phân chia tại một nút, một bài kiểm tra điều kiện được thực hiện dựa trên giá trị của một đặc trưng của tập con. Khi tập con chia sẻ cùng một nhãn lớp, hoặc không còn sự phân chia nào có thể cải thiện độ tinh khiết của lớp của tập con này, việc phân chia đệ quy tại nút này được coi là hoàn thành.

Về lý thuyết, đối với việc phân chia trên một đặc trưng (số hoặc hạng mục) với n giá trị khác nhau, có n cách khác nhau để phân chia nhị phân (có hoặc không đối với bài kiểm tra điều kiện), chưa kể đến các cách phân chia khác. Không tính đến thứ tự các đặc trưng mà việc phân chia diễn ra, đã có n^m cây có thể có cho một tập dữ liệu m-chiều.



Nhiều thuật toán đã được phát triển để xây dựng một cây quyết định chính xác một cách hiệu quả. Những thuật toán phổ biến bao gồm:

- ID3 (Iterative Dichotomiser 3): sử dụng một phương pháp tìm kiếm tham lam theo cách từ trên xuống bằng cách chọn thuộc tính tốt nhất để chia dữ liệu trên mỗi lần lặp và có khả năng quay lui khi cần thiết.
- C4.5: là phiên bản cải tiến của ID3 bằng cách giới thiệu khả năng quay lui khi duyệt cây và thay thế các nhánh bằng lá nếu độ tinh khiết được cải thiện theo cách này.
- CART (Classification and Regression Tree): sẽ được thảo luận chi tiết hơn.
- CHAID (Chi-squared Automatic Interaction Detector): thường được sử dụng trong thực tiễn tiếp thị trực tiếp. Nó bao gồm các phép tính thống kê phức tạp, nhưng cơ bản là xác định cách tối ưu để kết hợp các biến dự đoán để giải thích kết quả một cách tốt nhất.

Ý tưởng cơ bản của các thuật toán này là phát triển cây một cách tham lam bằng cách thực hiện một loạt các lựa chọn tối ưu về thuộc tính quan trọng để phân chia dữ liệu. Dữ liệu sau đó được chia dựa trên giá trị tối ưu của thuộc tính đó. Chúng ta sẽ thảo luận về việc đo lường các thuộc tính quan trọng và giá trị chia tối ưu của một thuộc tính trong phần tiếp theo.

Chúng ta sẽ nghiên cứu chi tiết và triển khai thuật toán CART như là thuật toán cây quyết định đáng chú ý nhất nói chung. Nó xây dựng cây bằng cách chia nhị phân liên tục và tìm kiếm thuật toán chia cây quyết định nổi bật nhất nói chung. Nó xây dựng cây bằng cách chia nhị phân liên tục và tìm kiếm sự kết hợp tối ưu của các thuộc tính và giá trị của chúng, nơi mà hàm đo lường chất lượng chia nhất định tìm kiếm các sự kết hợp có thể có của các thuộc tính và giá trị của chúng, nơi mà hàm đo lường chất lượng chia nhất định tìm kiếm các sự kết hợp có thể có của các thuộc tính và giá trị của chúng, và sau đó lựa chọn sự kết hợp tối ưu dựa trên các giá trị đo lường khác nhau. Với giá trị và thuộc tính được chọn làm điểm chia, nó chia dữ liệu theo cách sau:

- Mẫu với giá trị thuộc tính này (đối với thuộc tính phân loại) hoặc giá trị lớn hơn (đối với thuộc tính số) trở thành nút con bên phải.
- Các mẫu còn lại trở thành nút con bên trái.
-
- Quá trình phân chia trước đó lặp lại và chia đệ quy dữ liệu đầu vào thành hai nhóm con. Khi dữ liệu trở nên không hỗn hợp, quá trình chia dừng lại tại một nhóm con khi hai tiêu chí sau được đáp ứng:
-
- Số lượng mẫu tối thiểu cho một nút mới: Khi số lượng mẫu tại một nút không lớn hơn số lượng mẫu tối thiểu yêu cầu cho một nút con, quá trình phân chia dừng lại để ngăn chặn cây từ việc trở nên quá cụ thể.
- Độ sâu tối đa của cây: Một nút dừng phát triển khi độ sâu của nó, tính từ nút gốc, không nhỏ hơn độ sâu tối đa được xác định trước. Cây sâu hơn là cụ thể hơn cho dữ liệu huấn luyện và dẫn đến hiện tượng quá khớp.

Một nút không có nhánh ra trở thành một lá và lớp chiếm ưu thế của các mẫu tại nút này được sử dụng làm dự đoán. Khi quá trình phân chia kết thúc, thuật toán sẽ không có nhánh ra và lớp chiếm ưu thế của các mẫu tại nút này được sử dụng làm dự đoán. Khi quá trình phân chia kết thúc, thuật toán sẽ không có nhánh ra và lớp chiếm ưu thế của các mẫu tại nút này được sử dụng làm dự đoán. Khi quá trình phân chia kết thúc, thuật toán sẽ không có nhánh ra và lớp chiếm ưu thế của các mẫu tại nút này được sử dụng làm dự đoán. Khi quá trình phân chia kết thúc, chúng ta sẽ triển khai thuật toán cây quyết định CART từ đầu sau khi nghiên cứu các chỉ số chọn lựa các thuộc tính và giá trị chia tối ưu như đã hứa.

Các số liệu để đo lường sự phân chia

Khi lựa chọn sự kết hợp tốt nhất của các đặc trưng và giá trị làm điểm phân chia, hai tiêu chí là độ không tinh khiết Gini và lợi ích thông tin có thể được sử dụng để đo lường chất lượng của sự phân chia.

Độ không tinh khiết Gini, như tên gọi của nó, đo lường tỷ lệ không tinh khiết của lớp, tỷ lệ pha trộn của lớp. Đối với một tập dữ liệu với K lớp, giả sử dữ liệu từ lớp k (

$$1 \le k \le K$$

chiếm một phần nhỏ

$$f_k$$

(

$$0 \le f_k \le 1$$

) của toàn bộ tập dữ liệu, tạp chất Gini của tập dữ liệu đó được viết như sau:

$$Gini\ impurity = 1 - \sum_{k=1}^{K} f_k^2$$

Chỉ số Gini thấp chỉ ra rằng tập dữ liệu sạch hơn. Ví dụ, khi tập dữ liệu chỉ chứa một lớp, giả sử tỷ lệ của lớp này là 1 và của các lớp khác là 0, chỉ số Gini của nó sẽ là 1 - (1^2 + 0^2) = 0. Trong một ví dụ khác, một tập dữ liệu ghi lại một số lượng lớn lần tung đồng xu nơi mà mặt ngửa và mặt sấp chiếm mỗi nửa số mẫu, chỉ số Gini là:

$$1 - (0.5^2 + 0.5^2) = 0.5$$

Trong trường hợp nhị phân, độ không hoàn hảo Gini dưới các giá trị khác nhau của phần tử lớp tích cực có thể được trực quan hóa bởi đoạn mã sau:

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
```

Một phần của lớp tích cực thay đổi từ 0 đến 1:

```
>>> pos_fraction = np.linspace(0.00, 1.00, 1000)
```