SVM

Do you smell what's SVM is cooking?

Files: http://bit.ly/2kOA6xY

Score board: http://bit.ly/2jOAejv

Alexandr Notchenko

Alexandr.Notchenko@skolkovotech.ru

Plan

- Quickly re-iterating terminology and concepts about SVM from lecture.
- Checking out (L-)SVC and SVR functions in sklearn.svm , discussing their signatures and arguments
- Looking into what's cooking kaggle competition, Exploring dataset quickly
  - Thinking how can you convert text data into vectors for SVC input
  - discussing Using OneHotEncoding vs TfIdfEncoder
  - 5-fold stratified validation split
  - Talking about properties of OneVsAll classification setup
  - Training final evaluating models
  - Plotting results
  - Changing hyper-parameters C (margin) and d (degree of polynomial) and plo corresponding change in performance
- Finishing words

Reiterating terminology and concepts

Supervised Learning

- **Training data**: sample S of size m drawn $i.i.d.$ according to distribution D on $X \times Y$

$$S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$$

- **Problem** find hypothesis $h \in H$ with small generalization error
  - deterministic case: $y = f(x)$ is a deterministic function, only $x \sim D$
  - stochastic case: output is a probabilistic function of input, e.g. $y = f(x) + \varepsilon$

`sklearn.svm`: Support Vector Machines

The `sklearn.svm` module includes Support Vector Machine algorithms.

User guide: See the Support Vector Machines section for further details.

Estimators

`svm.SVC` ([C, kernel, degree, gamma, coef0, ...])	C-Support Vector Classification.
`svm.LinearSVC` ([penalty, loss, dual, tol, C, ...])	Linear Support Vector Classification.
`svm.NuSVC` ([nu, kernel, degree, gamma, ...])	Nu-Support Vector Classification.
`svm.SVR` ([kernel, degree, gamma, coef0, tol, ...])	Epsilon-Support Vector Regression.
`svm.LinearSVR` ([epsilon, tol, C, loss, ...])	Linear Support Vector Regression.
`svm.NuSVR` ([nu, C, kernel, degree, gamma, ...])	Nu Support Vector Regression.
`svm.OneClassSVM` ([kernel, degree, gamma, ...])	Unsupervised Outlier Detection.
`svm.l1_min_c` (X, y[, loss, fit_intercept, ...])	Return the lowest bound for C such that for C in (l1_min_C, infinity) the model is guaranteed not to be empty.

Low-level methods

`svm.libsvm.fit`	Train the model using libsvm (low-level method)
`svm.libsvm.decision_function`	Predict margin (libsvm name for this is predict_values)
`svm.libsvm.predict`	Predict target values of X given a model (low-level method)
`svm.libsvm.predict_proba`	Predict probabilities
`svm.libsvm.cross_validation`	Binding of the cross-validation routine (low-level routine)

class `sklearn.svm.` **SVC** (*C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape=None, random_state=None*) [source]

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

C : float, optional (default=1.0) - Penalty parameter C of the error term

kernel : string, optional (default='rbf') - kernel type ('linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable)

degree : int, optional (default=3) - Degree of the polynomial kernel function ('poly')

gamma : float, optional (default='auto', =1/n_features) - Precision for 'rbf', or multiplier for 'poly' and 'sigmoid'

coef0 : float, optional (default = 0) - Constant terms for 'poly' and 'sigmoid' kernels

class_weight : {dict, 'balanced'}, optional - Set the parameter C of class i to class_weight[i]*C for SVC. If not given, all classes are supposed to have weight one. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y))

decision_function_shape : 'ovo', 'ovr' or None, default=None - Whether to return a one-vs-rest ('ovr') decision function of shape (n_samples, n_classes) as all other classifiers, or the original one-vs-one ('ovo') decision function of libsvm which has shape (n_samples, n_classes * (n_classes - 1) / 2)

tol : float, optional (default=1e-3) - Tolerance for stopping criterion (affects solution accuracy)

class `sklearn.svm.` **`LinearSVC`** (*penalty='l2', loss='squared_hinge', dual=True, tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000*) [source]

http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC

C : float, optional (default=1.0) - Penalty parameter C of the error term.

loss : string, 'hinge' or 'squared_hinge' (default='squared_hinge') - Specifies the loss function. 'hinge' is the standard SVM loss (used e.g. by the SVC class) while 'squared_hinge' is the square of the hinge loss.

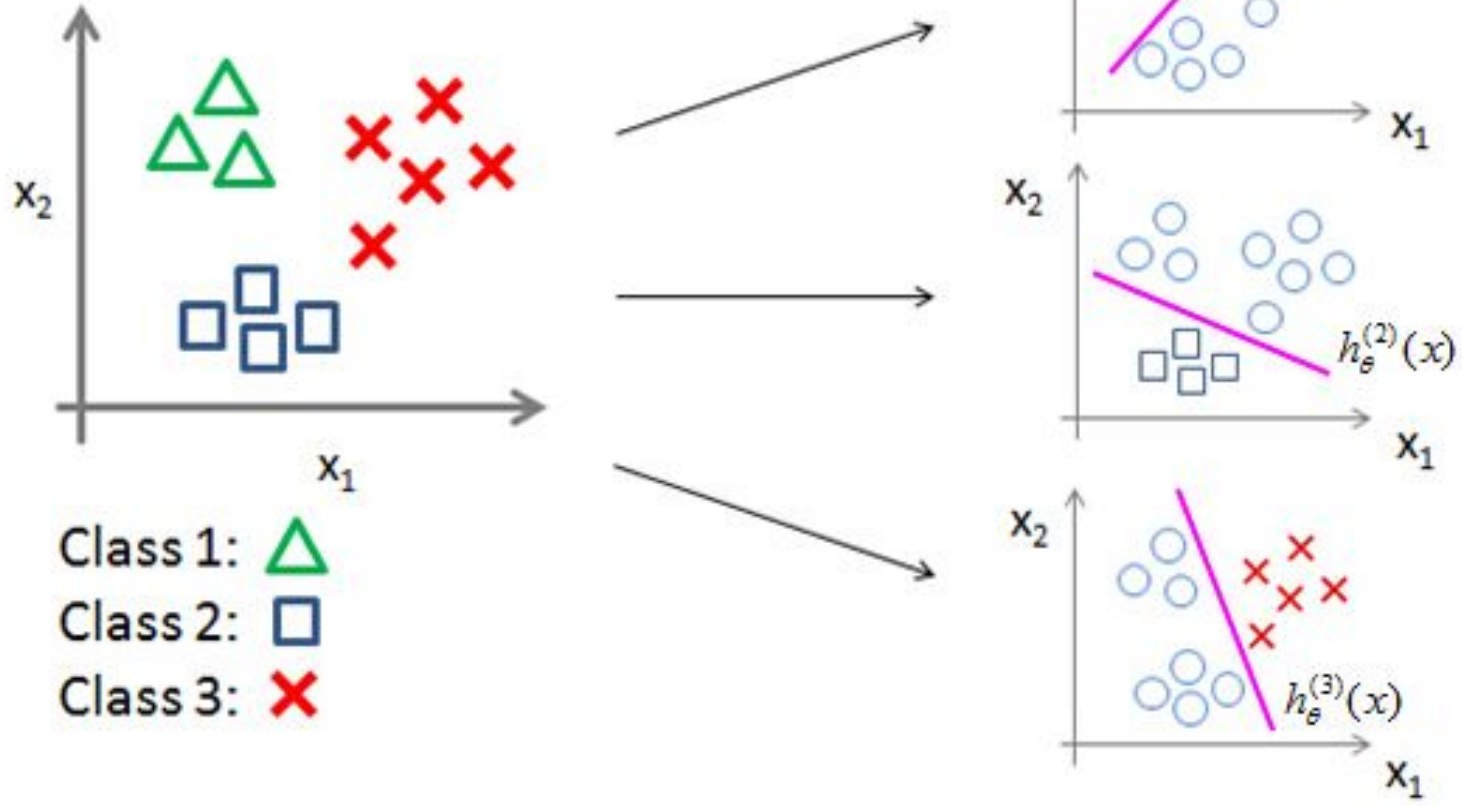
penalty : string, 'l1' or 'l2' (default='l2') - Specifies the norm used in the penalization. The 'l2' penalty is the standard used in SVC. The 'l1' leads to coef_ vectors that are sparse.

tol : float, optional (default=1e-4) - Tolerance for stopping criteria.

multi_class: string, (default='ovr') : - Determines the multi-class strategy if y contains more than two classes.

class_weight : {dict, 'balanced'}, optional - Set the parameter C of class i to class_weight[i]*C for SVC.

One-vs-rest classifier



What’s cooking dataset

https://www.kaggle.com/c/whats-cooking

Completed • Knowledge • 1,388 teams

What's Cooking?

Wed 9 Sep 2015 – Sun 20 Dec 2015 (13 months ago)

Data Files

File Name	Available Formats
sample_submission.csv	.zip (25.80 kb)
test.json	.zip (425.52 kb)
train.json	.zip (1.76 mb)


```
{
"id": 24717,
"cuisine": "indian",
"ingredients": [
    "tumeric",
    "vegetable stock",
    "tomatoes",
    "garam masala",
    "naan",
    "red lentils",
    "red chili peppers",
    "onions",
    "spinach",
    "sweet potatoes"
]
},
```

How can we convert text data into vectors for SVC?

- Sequences of words?
- Interaction of words on feature level or before?
- Can Stemming help?
- N-grams?

One-hot encoding

```
V = {zebra, horse, school, summer}
```

```
v(zebra) = [1, 0, 0, 0]
v(horse) = [0, 1, 0, 0]
v(school) = [0, 0, 1, 0]
v(summer) = [0, 0, 0, 1]
```

(+) Pros:

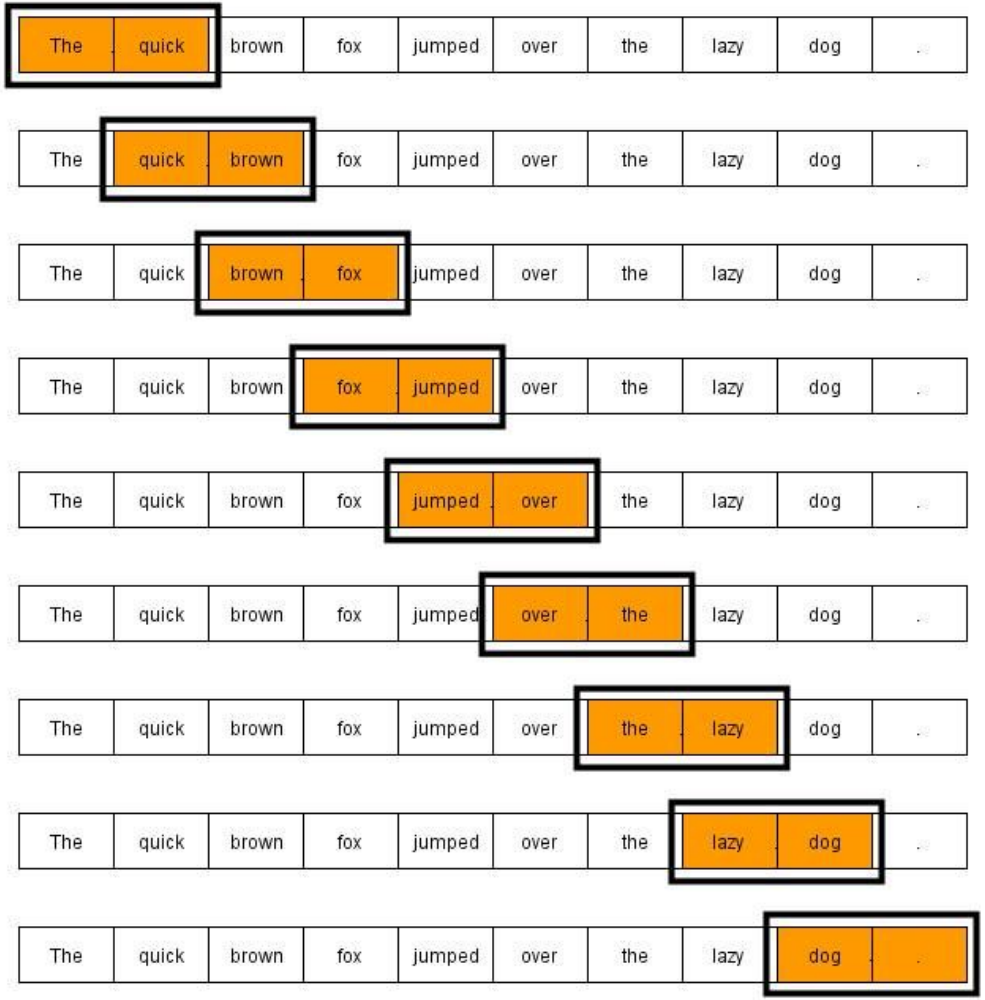
Simplicity

(-) Cons:

One-hot encoding can be memory inefficient

Notion of word similarity is undefined with one-hot encoding

n-grams


The quick brown fox jumped over the lazy dog .
The quick brown fox jumped over the lazy dog .
The quick brown fox jumped over the lazy dog .
The quick brown fox jumped over the lazy dog .
The quick brown fox jumped over the lazy dog .
The quick brown fox jumped over the lazy dog .
The quick brown fox jumped over the lazy dog .
The quick brown fox jumped over the lazy dog .
The quick brown fox jumped over the lazy dog .

TFIDF

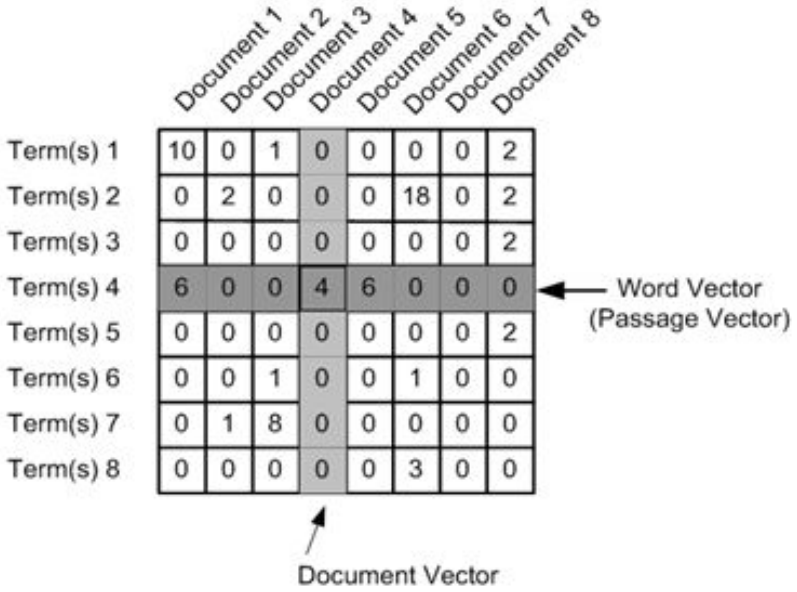
For a term i in document j:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j

df_i = number of documents containing i

N = total number of documents



class `sklearn.feature_extraction.text.` **`TfidfTransformer`** (*norm=u'l2', use_idf=True, smooth_idf=True, sublinear_tf=False*) [source]

Sources*:

1. NYU ML Lectures - Support Vector Machines
2. NICTA MLSS - Classification notebook
3. Advanced BioStatistics - 8366 section 6.4
4. Support Vector Machine Tutorial - Wu, Shih-Hung (Ph.D) - Dept. of CSIE, CYUT
5. Michal Tadeusiak's solution
6. SaquibAhmad solution to What's cooking Kaggle competition

*(Notebooks and slides from which to take some code/pictures/equations)