Text Classification with Support Vector Machines

Natural Language Processing Laboratory
2017/04/11

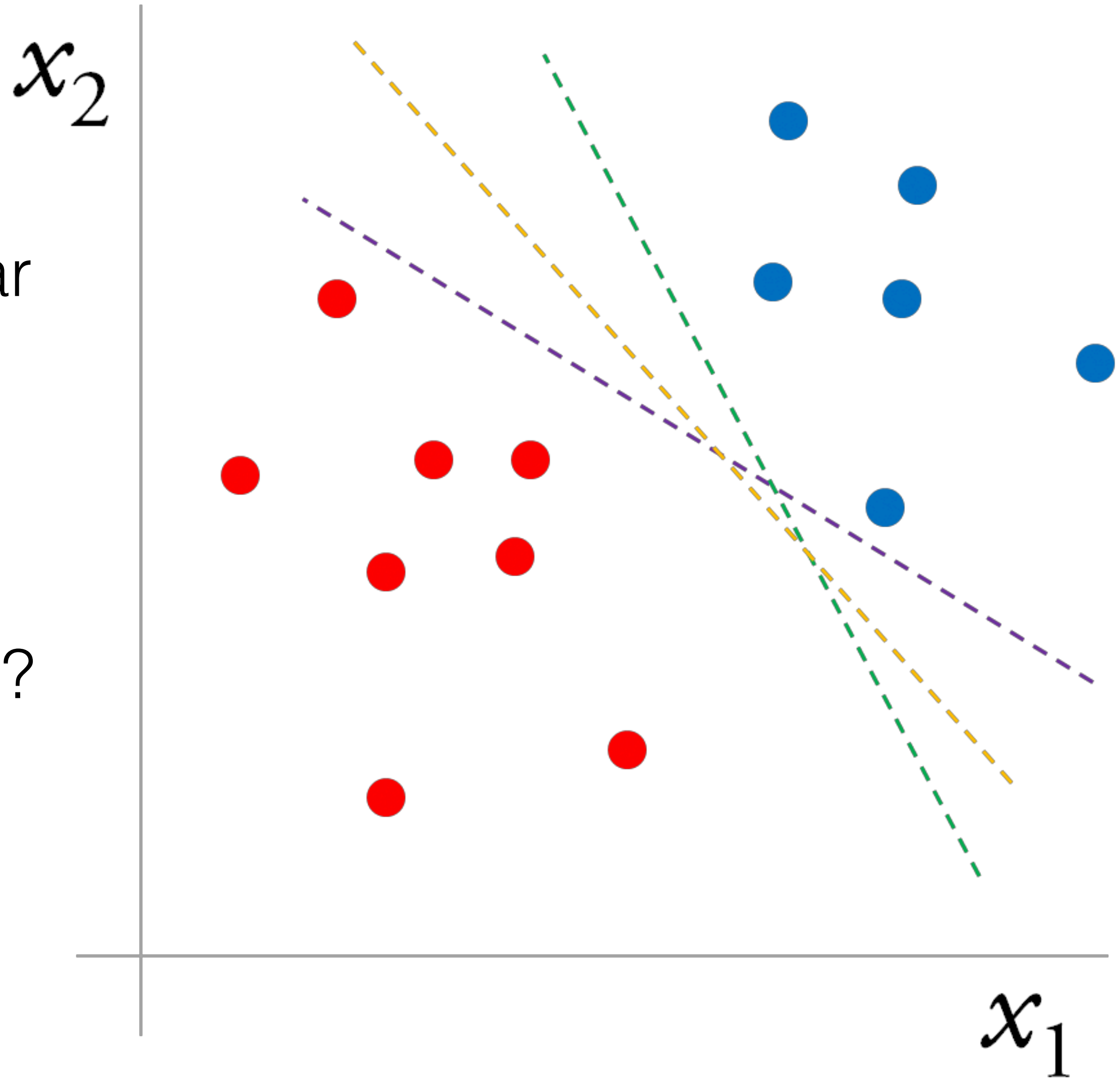
Support Vector Machines

- A set of supervised learning methods
  - Used for classification, regression and outliers detection.
- Probably the best “off-the-shelf” supervised learning algorithm

Linear Discriminant Function

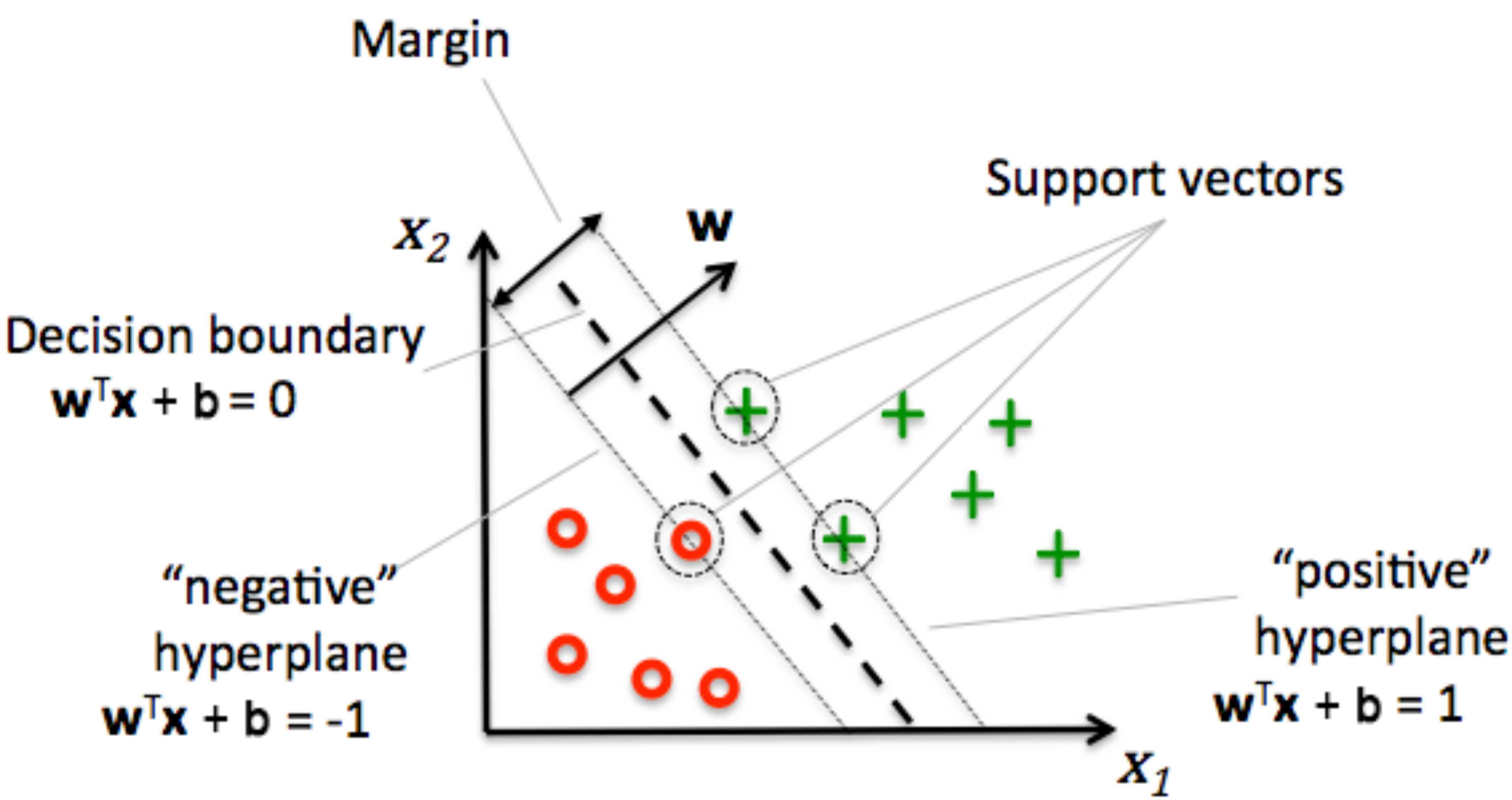
$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

- There are many feasible linear discriminant functions when the classes are linearly separable
- Which hyperplane is the best?

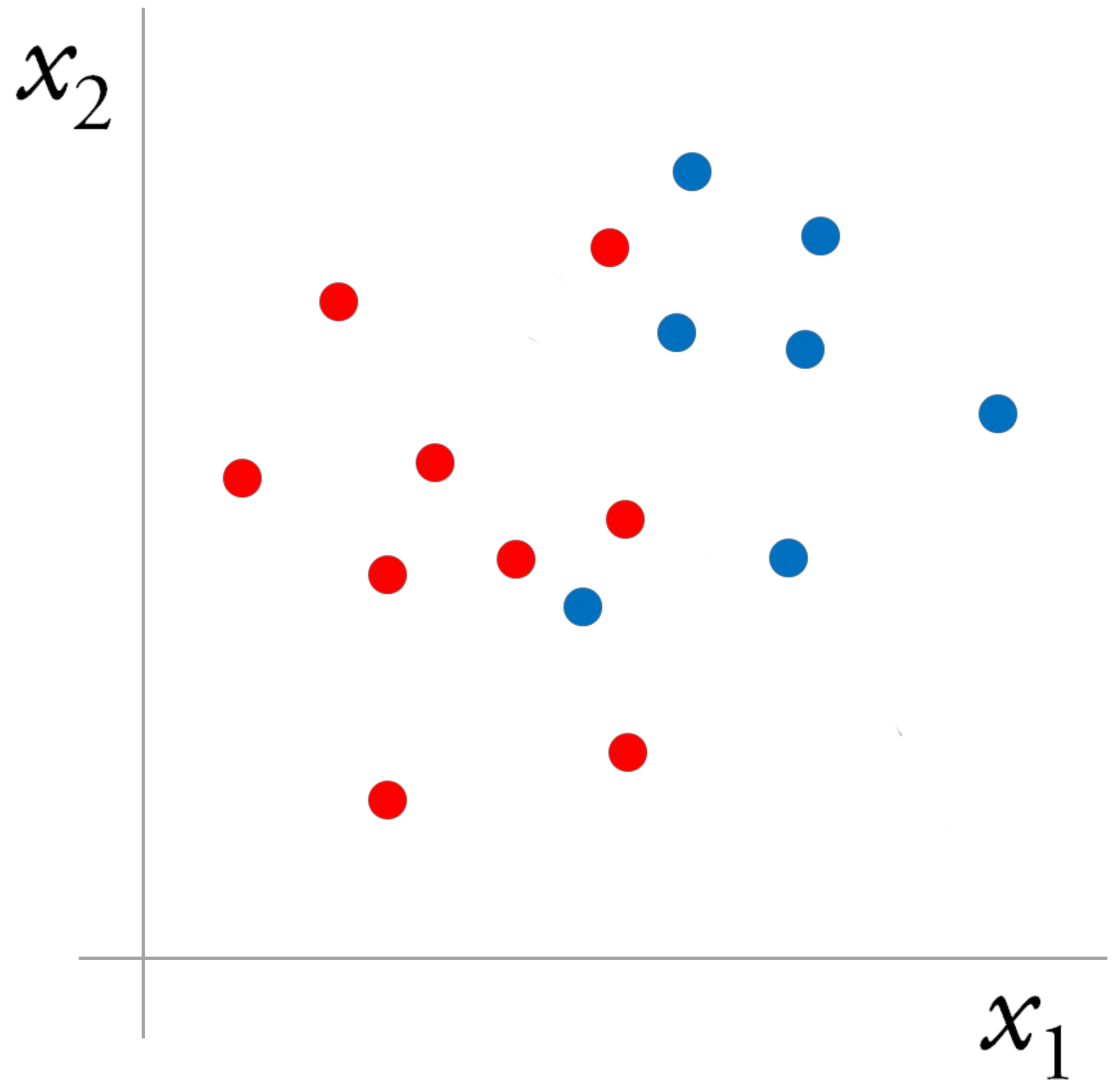


Support Vector Classification

- Support vector classifier (SVC) picks one with largest margin



What if the data is not linear separable?



Slack variables ξ_i

- Tolerate slacks that fall outside of the regions

- Formulation

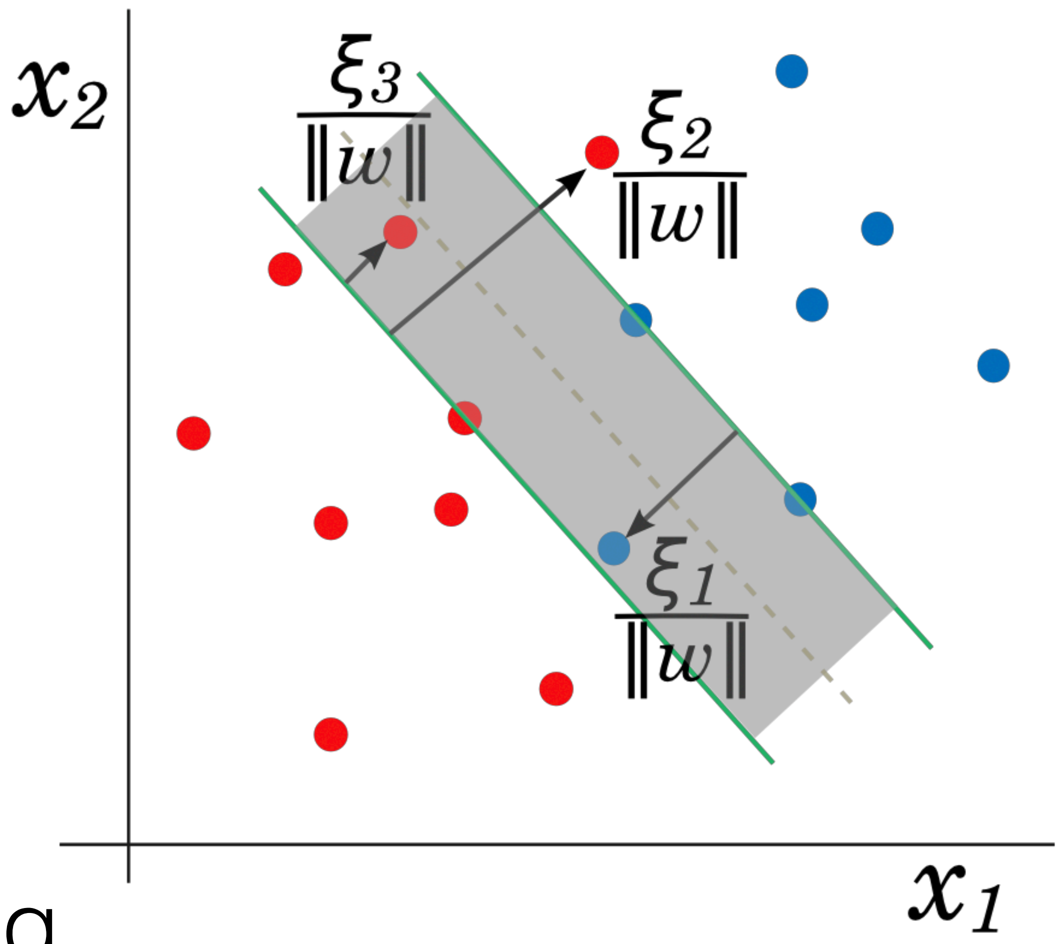
$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

- Such that

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

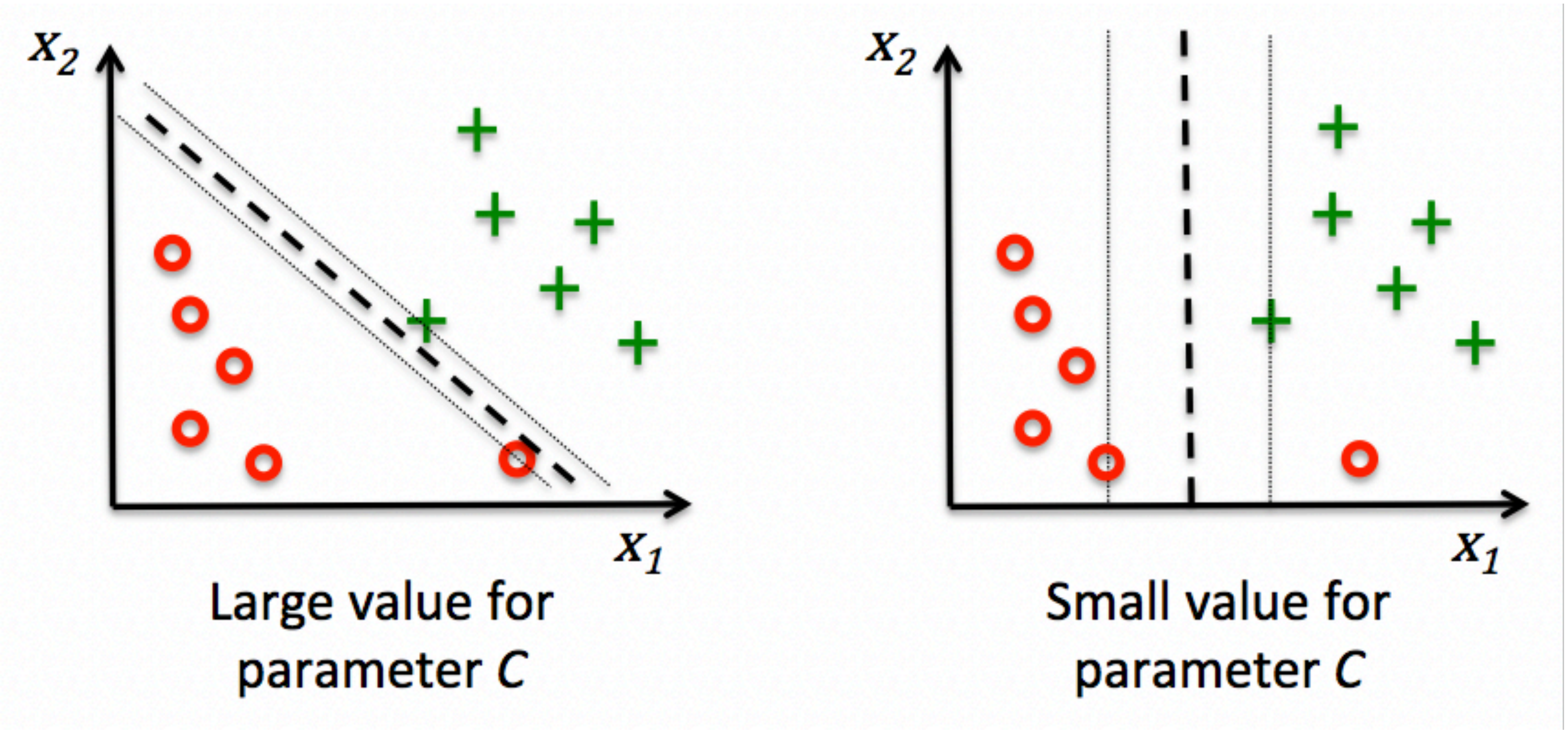
$$\xi_i \geq 0$$

- Parameter C can be viewed as a way to control over-fitting



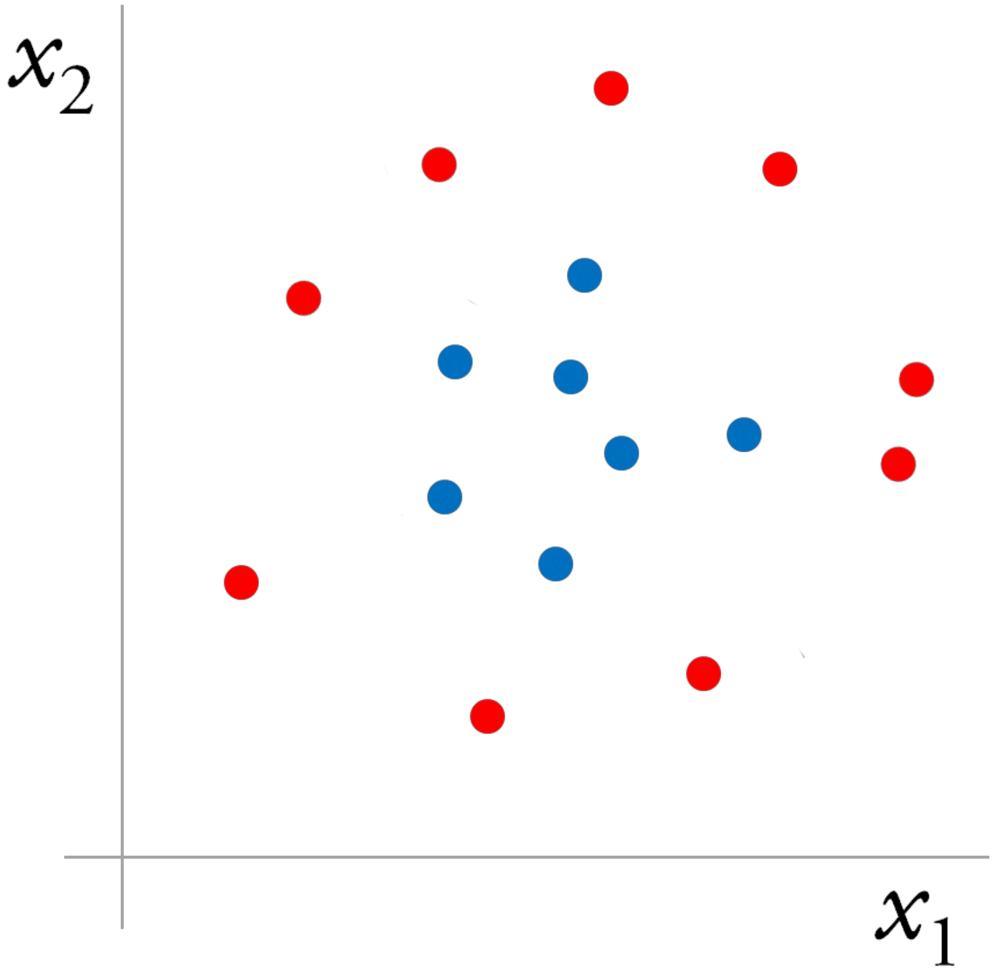
Tradeoff between C and Margin

- Hyperparameter C controls the tradeoff between
  - Maximizing margin
  - Minimizing number of slacks



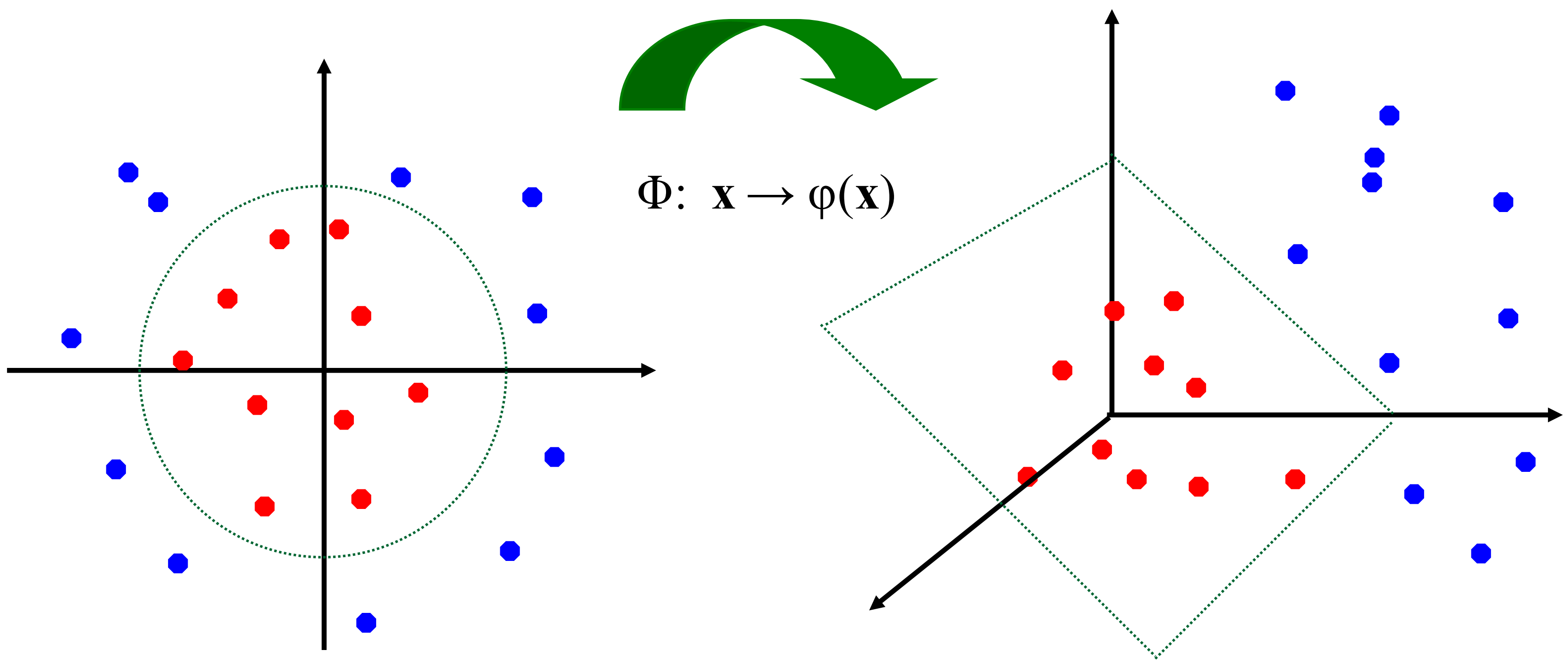
Nonlinearly Separable Classes

- In practice, classes may be nonlinearly separable

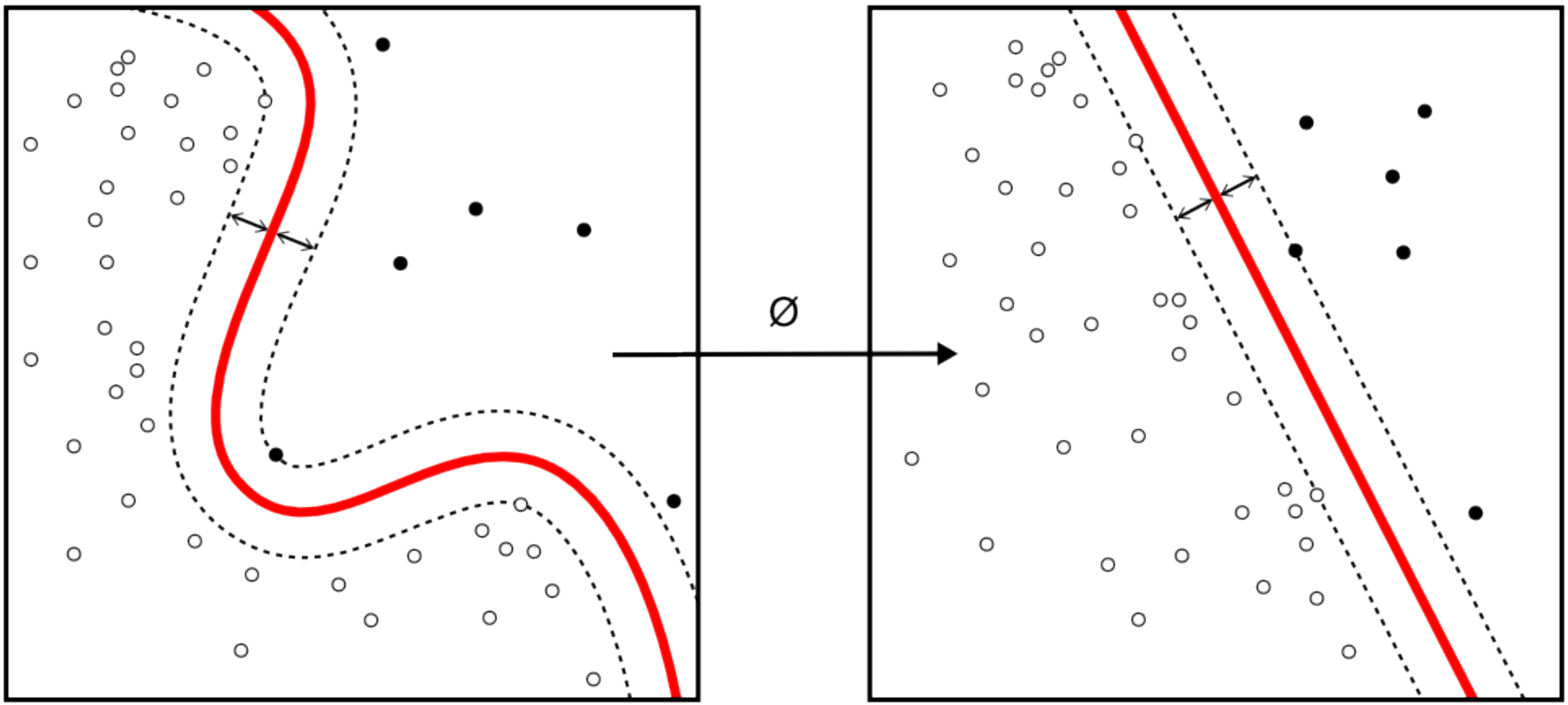


Feature Space Transformation

- Map to some higher-dimensional feature spaces where the training set is separable



Kernel Trick

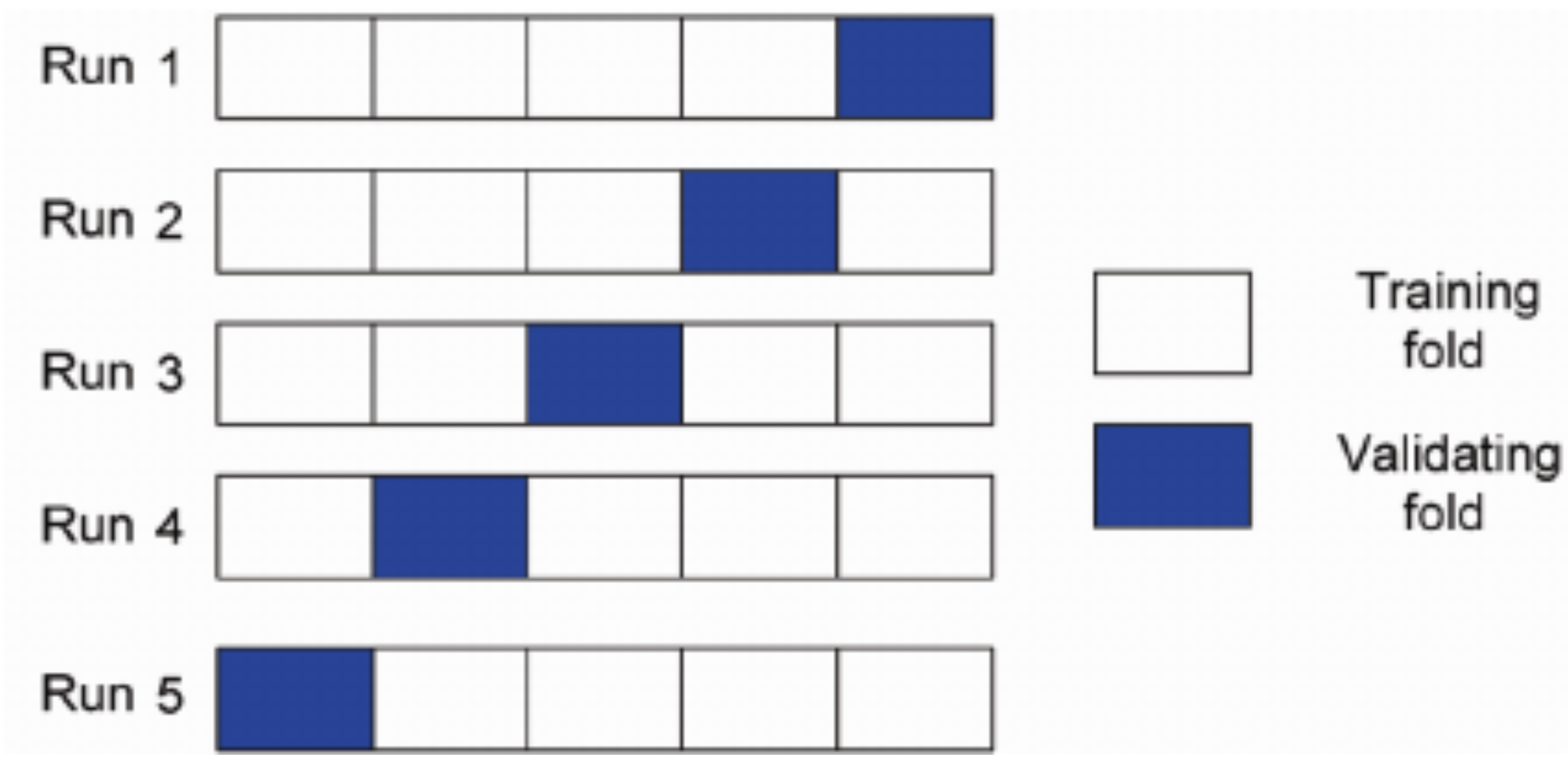


https://en.wikipedia.org/wiki/Support_vector_machine

Hyperparameter Tuning

- Hyperparameter combination (C,γ)
- Try out all possible combinations exhaustively
- This procedure is called the *grid search*

Cross Validation



Multiclass Classification

- SVM is inherently binary
- One-against-one
  - (a, b), (b, c), (a, c)
  - n*(n-1)/2 classifiers
- One-against-the-rest
  - (a, (b or c)), (b, (a or c)), (c, (a, b))
  - n classifiers

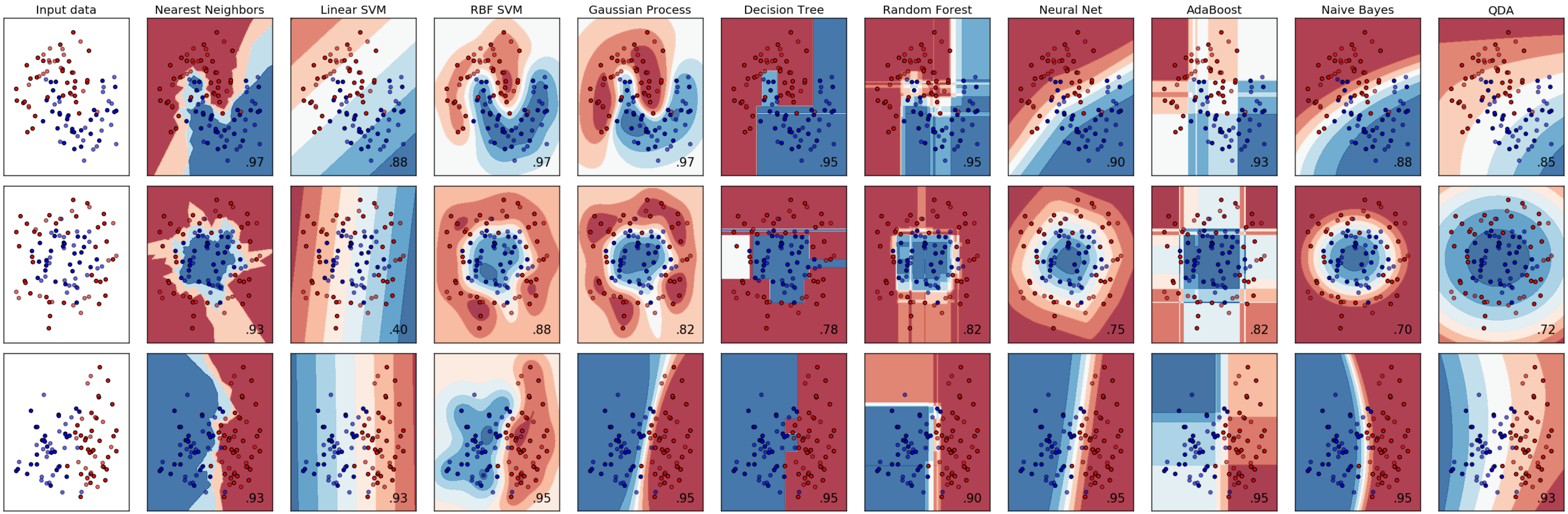
Pros & Cons

- Effective in high dimensional spaces.
- Memory efficient
  - Only uses a subset of training points in the decision function (called *support vectors*)
- Versatile
  - *Kernel functions* can be specified for the decision function.

- Did not provide probability estimates

http://scikit-learn.org/stable/modules/svm.html

Classifier Comparison



http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Working With Text Data

- Text are strings of characters

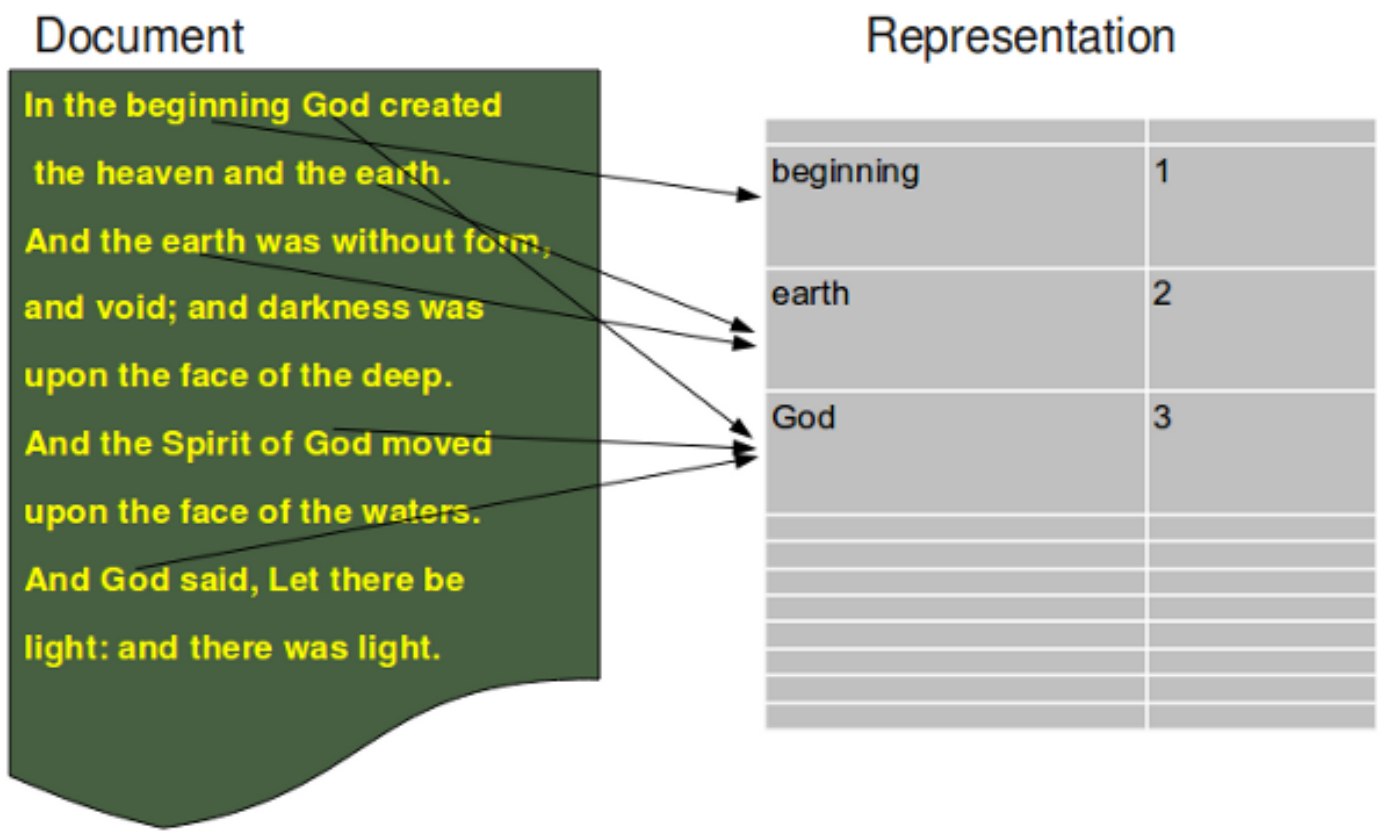
- Transform documents into a representation suitable for the learning algorithm

- What’s the representation of words, sentences, or documents?

Extract Features from Text

- Bag of words
- TF-IDF
- word2vec (will introduce next week)

Bag of words



- Continuous Bag of Words (CBOW)
- Skip-gram

tf-idf

- Short for **term frequency–inverse document frequency**

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

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

tf-idf

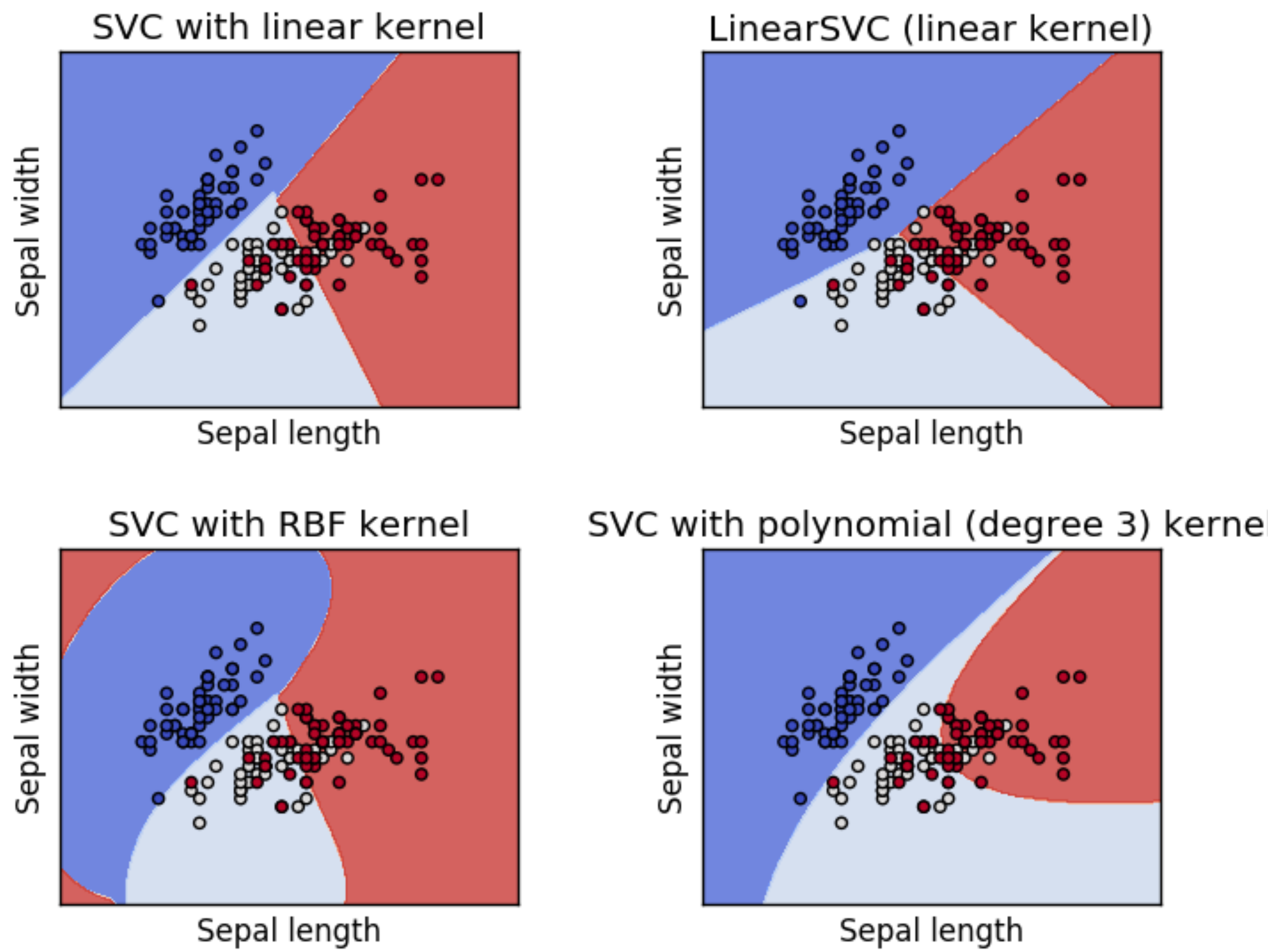
- TF: Term Frequency
    - Proportion to a document
    - need normalize
- IDF: Inverse Document Frequency
    - Different words have different significance

Play SVM with scikit-learn

- In *sklearn.svm* module
  - SVC (libsvm)
  - LinearSVC (liblinear)
    Similar to SVC with kernel='linear', but more flexible in the choice of penalties and loss functions and scale better to large numbers of samples
  - NuSVC
    Similar to SVC but uses a parameter to control *#support vectors*
- All capable of performing multi-class classification
  - one-vs-one: SVC, NuSVC
  - one-vs-the-rest: LinearSVC (support crammer_singer)

http://scikit-learn.org/stable/modules/svm.html

Kernels You Can Use



or self-defined kernel

http://scikit-learn.org/stable/modules/svm.html

Linear SVC

```
from sklearn.svm import SVC

# kernel: kernel can be 'linear', 'poly', 'rbf', ...etc
# C is the hyperparameter for the error penalty term
svm_linear = SVC(kernel='linear', C=1000.0,
random_state=0)

svm_linear.fit(X_train, y_train)
y_pred = svm_linear.predict(X_test)
```

RBF SVC

```python
from sklearn.svm import SVC

# C is the hyperparameter for the error penalty term
# gamma is the hyperparameter for the rbf kernel
svm_rbf = SVC(kernel='rbf', random_state=0, gamma=0.2, C=10.0)

svm_rbf.fit(X_train, y_train)
y_pred = svm_rbf.predict(X_test)
```

Tuning Hyperparameters via Grid Search

```
from sklearn.model_selection import GridSearchCV

param_C = [0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0]
param_gamma = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0]

svm = SVC(random_state=0)

# set the parameter of GridSearchCV to a list of dictionaries
param_grid = [{'C': param_C,
               'gamma': param_gamma,
               'kernel': ['rbf']}]
gs = GridSearchCV(estimator=svm,
                  param_grid=param_grid,
                  scoring='accuracy')

gs = gs.fit(X_train, y_train)
print(gs.best_score_, gs.best_params_)
```

Use Tuned Classifier

```
# get the best estimator
clf = gs.best_estimator_
# train the data
clf.fit(X_train, y_train)

# test accuracy
clf.score(X_test_std, y_test)
```

Extract Text Features

- CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(twenty_train.data)

count_vect.vocabulary_.get(u'algorithm')

# ngram count
ngram_vect = CountVectorizer(ngram_range=(1, 5))
X_train_counts = ngram_count_vect.fit_transform(twenty_train.data)

count_vect.vocabulary_.get(u’algorithm for')
```

Extract Text Features

- TFIDF Transformer

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
```

- TFIDF Vectorizer

  - Equivalent to CountVectorizer followed by TfidfTransformer

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(twenty_train.data)
```