Lab 6

Support Vector Machines

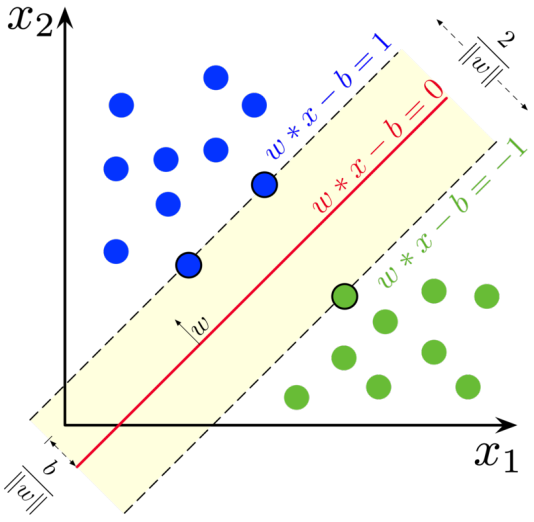
Introduction

- Objectives
  - SPAM dataset: a binary classification problem
  - Linear and non-linear Support Vector Machine classifiers
  - Hyperparameter selection
  - Performance metrics

SVM

Linear SVM classifier:

Separable classes: An SVM classifier seeks the hyperplane that best separates samples from the two classes



Function to minimize:

$$L = \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i \left(\mathbf{w}^T \mathbf{x}_i + w_0 \right) - 1 \right)$$

We obtain a convex problem depending on α_i. It can be solved using standard optimization software

$$L = \sum_{i=1}^{N} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k \mathbf{x}_i^T \mathbf{x}_k \qquad \text{subject to} \quad \begin{cases} \alpha_i \geq 0 \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases}$$

Classification of a vector x

$$\hat{y} = sign\left(g(\mathbf{x}) \right) = sign\left(\mathbf{w}^T \mathbf{x} + w_0 \right) = sign\left(\sum_{i=1}^{N_{SV}} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + w_0 \right)$$

SVM

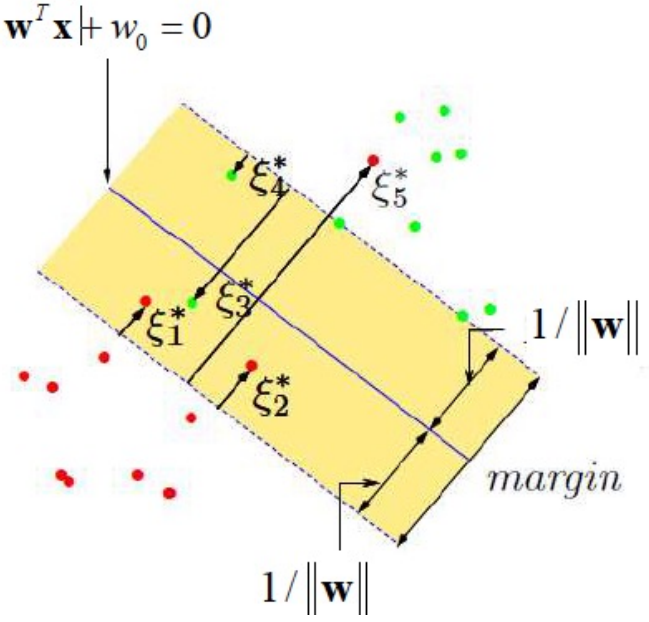
Linear SVM classifier:

Non-separable classes: no hyperplane can separate the classes without error. We permit some training vectors wrongly classified

$$y_i\left(\mathbf{w}^T\mathbf{x}_i + w_o\right) \geq 1-\xi_i \qquad i=1,\ldots,N$$

and introduce a penalization for non-null values of ξ_i :

$$\underset{\xi,\mathbf{w},w_o}{\text{minimize}} \ \tfrac{1}{2}\|\mathbf{w}\|^2 + P\sum_{i=1}^{N}\xi_i \quad \text{subject to} \begin{cases} y_i\left(\mathbf{w}^T\mathbf{x}_i + w_o\right) \geq 1-\xi_i \\ \xi_i \geq 0 \end{cases} \quad i=1,\ldots,N$$



Using the Lagrangian:

$$L = \tfrac{1}{2}\|\mathbf{w}\|^2 + P\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\left(y_i\left(\mathbf{w}^T\mathbf{x}_i + w_0\right) - \left(1-\xi_i\right)\right) - \sum_{i=1}^{N}\beta_i\xi_i$$

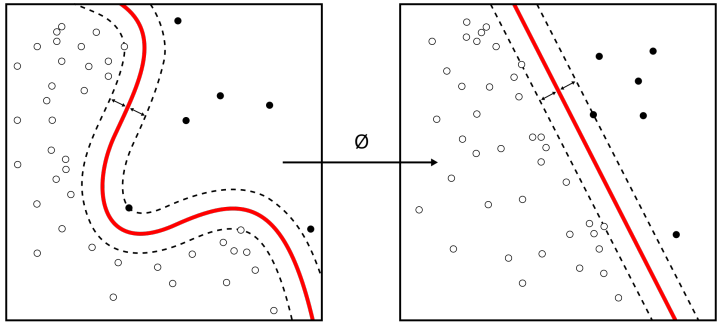
The penalization parameter P must be validated

SVM

Non-linear SVM classifier:

Uses kernel functions

$$L = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k K(\mathbf{x}_i, \mathbf{x}_k) \quad \text{subject to} \quad \begin{cases} 0 \le \alpha_i \le P \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases}$$



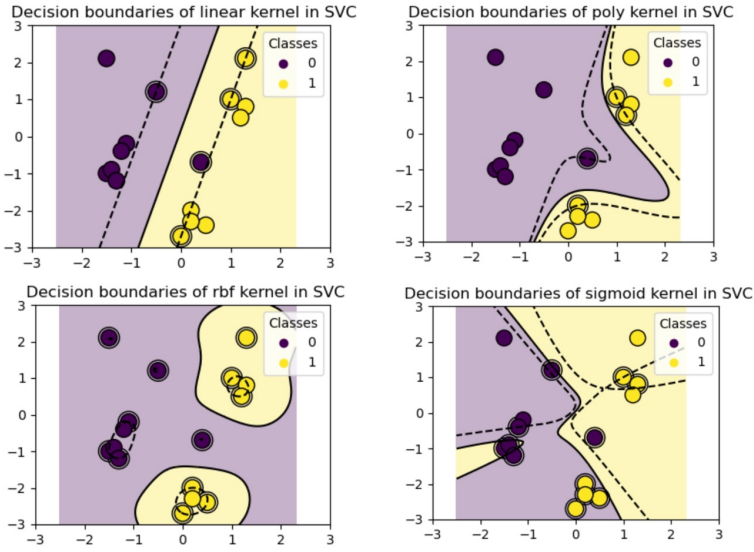
Example: a Gaussian Kernel

$$K(\mathbf{x}_i, \mathbf{x}_k) = \exp\left(-\frac{1}{\sigma^2} \|\mathbf{x}_i - \mathbf{x}_k\|^2\right)$$

σ^2 is a parameter that must be validated

Classification of a vector **x**:

$$\hat{y} = sign\left(\sum_{i=1}^{N_{SV}} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + w_0\right)$$



SPAM dataset

Each vector in the dataset corresponds to a received email

Dataset:

- Classes: c=2 (spam, mail)
- Samples: N = 4601 (1813 spam and 2788 mail)
- Features: d=57 frequency of a particular word in the email. The last features correspond to run-length attributes that measure the length of sequences of consecutive capital letters.

Goal: build a spam filter

Content of a feature vector

Number	Feature	Number	Feature
1	word_freq_make: continuous.	30	word_freq_labs: continuous.
2	word_freq_address: continuous.	31	word_freq_telnet: continuous.
3	word_freq_all: continuous.	32	word_freq_857: continuous.
4	word_freq_3d: continuous.	33	word_freq_data: continuous.
5	word_freq_our: continuous.	34	word_freq_415: continuous.
6	word_freq_over: continuous.	35	word_freq_85: continuous.
7	word_freq_remove: continuous.	36	word_freq_technology: continuous.
8	word_freq_internet: continuous.	37	word_freq_1999: continuous.
9	word_freq_order: continuous.	38	word_freq_parts: continuous.
10	word_freq_mail: continuous.	39	word_freq_pm: continuous.
11	word_freq_receive: continuous.	40	word_freq_direct: continuous.
12	word_freq_will: continuous.	41	word_freq_cs: continuous.
13	word_freq_people: continuous.	42	word_freq_meeting: continuous.
14	word_freq_report: continuous.	43	word_freq_original: continuous.
15	word_freq_addresses: continuous.	44	word_freq_project: continuous.
16	word_freq_free: continuous.	45	word_freq_re: continuous.
17	word_freq_business: continuous.	46	word_freq_edu: continuous.
18	word_freq_email: continuous.	47	word_freq_table: continuous.
19	word_freq_you: continuous.	48	word_freq_conference: continuous
20	word_freq_credit: continuous.	49	char_freq_;: continuous.
21	word_freq_your: continuous.	50	char_freq_(: continuous.
22	word_freq_font: continuous.	51	char_freq_[: continuous.
23	word_freq_000: continuous.	52	char_freq_!: continuous.
24	word_freq_money: continuous.	53	char_freq_$: continuous.
25	word_freq_hp: continuous.	54	char_freq_#: continuous.
26	word_freq_hpl: continuous.	55	capital_run_length_average: continu
27	word_freq_george: continuous.	56	capital_run_length_longest: continuc
28	word_freq_650: continuous.	57	capital_run_length_total: continuou
29	word_freq_lab: continuous.		

Validation of parameters

Dataset split into 2 subsets (stratified splitting)

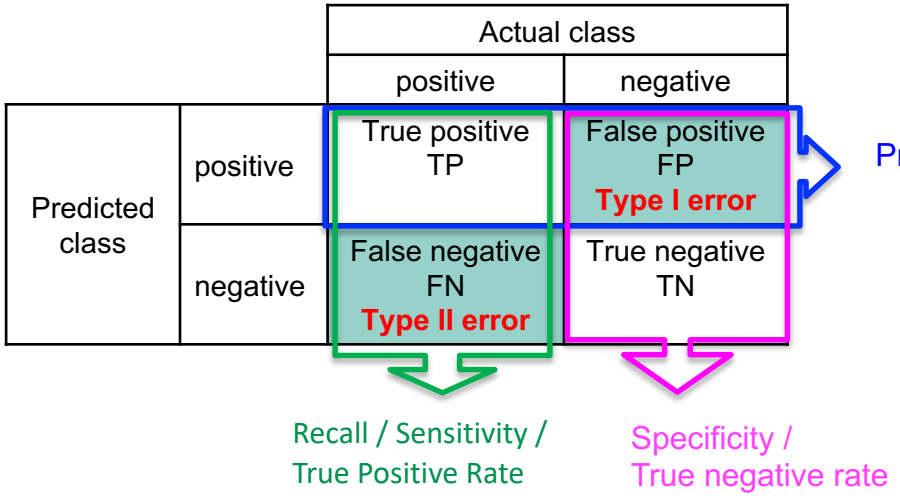
- Train: X_train, y_train, 80%
- Test: X_test, y_test, 20%

Parameter validation (brute force: grid search):

- Split Train set into training (75%) and validation (25%) subsets
- Train linear SVM
  - Find best hyperparameter P
- Train non-linear SVM
  - Find best hyperparameters (P, σ_2)
  - compute f1 score on validation set
- Compute metrics on the train / test set

Performance metrics

Precision, Recall (=Sensitivity), Specificity:



$$P = \frac{TP}{TP + FP} = \frac{\#\text{correctly classified as SPAM}}{\#\text{classified as SPAM}}$$

$$R = Sens = \frac{TP}{TP + FN} = \frac{\#\text{correctly classified as SPAM}}{\#\text{total of SPAM}}$$

$$Sp = \frac{TN}{TN + FP} = \frac{\#\text{correctly classified as MAIL}}{\#\text{total of MAIL}}$$

F score: A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score:

$$F_{score} = 2\frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}}$$

SVM in ScikitLearn

```
class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale',
 coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=2
00, class_weight=None, verbose=False, max_iter=1, decision_function_sh
ape='ovr', break_ties=False, random_state=None)
```

C ***:*** Regularization parameter. The strength of the regularization is inversely proportional to C

kernel*{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}* specifies the type of kernel

Be careful sklearn rbf parameter gamma

https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

```
from sklearn import svm
clf = svm.SVC()
clf.fit(X_train, y_train)
clf.predict(X_test)
```

SVM in ScikitLearn

Notes:

- The multiclass support is handled according to a one-vs-one scheme. In total, `n_classes * (n_classes - 1) / 2` classifiers are constructed and each one trains data from two classes.

- The implementation is based on `libsvm`. The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. For large datasets consider using `LinearSVC`

- Some properties of these support vectors can be found in attributes

  support_vectors_ get support vectors

  support_ get índices of support vectors

  n_support: get number of support vectors for each class

- In problems where it is desired to give more importance to certain classes or certain individual samples, the parameters `class_weight` can be used

- Prediction methods: `predict` or `predict_proba, predict_log_proba`

  predict For a one-class model, +1 or -1 is returned.

  proba methosds Compute probabilities of possible outcomes for samples in X. The model needs to have probability information computed at training time: fit with attribute probability set to True. The probability model is created using cross validation, so the results can be slightly different than those obtained by predict.