Fundamentals of machine learning

Course 5: linear classification

Elisa Riccietti & Titouan Vayer
email: elisa.riccietti@ens-lyon.fr, titouan.vayer@inria.fr

February 18, 2025

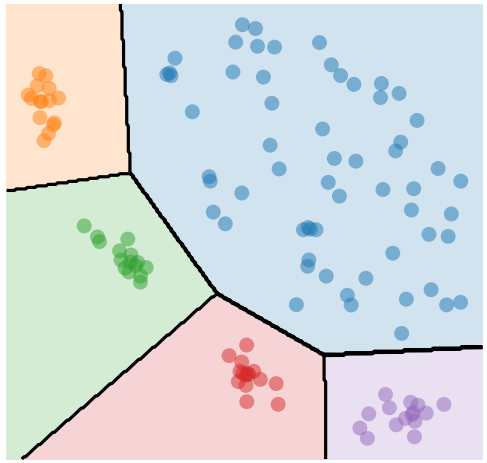
Linear methods for classification

Given data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ we want to assign each $\mathbf{x}_i$ to a class, we look for a function c such that

$$f : \mathbb{R}^d \to \{1, \ldots, K\}$$
$$\mathbf{x} \to y$$

Linear classification

A classification predictor c divides the space in K subregions (number of classes). The *decision boundaries* (boundaries of these regions) are linear



How to obtain linear boundaries? (I)

- Supervised linear fit in each class (linear *discriminant function*):

$$\hat{f}_k(\mathbf{x}) = \hat{\theta}_{k,0} + \hat{\boldsymbol{\theta}}_k^T \mathbf{x},\ k = 1, \ldots, K$$

- The *decision boundary* between class k and ℓ:

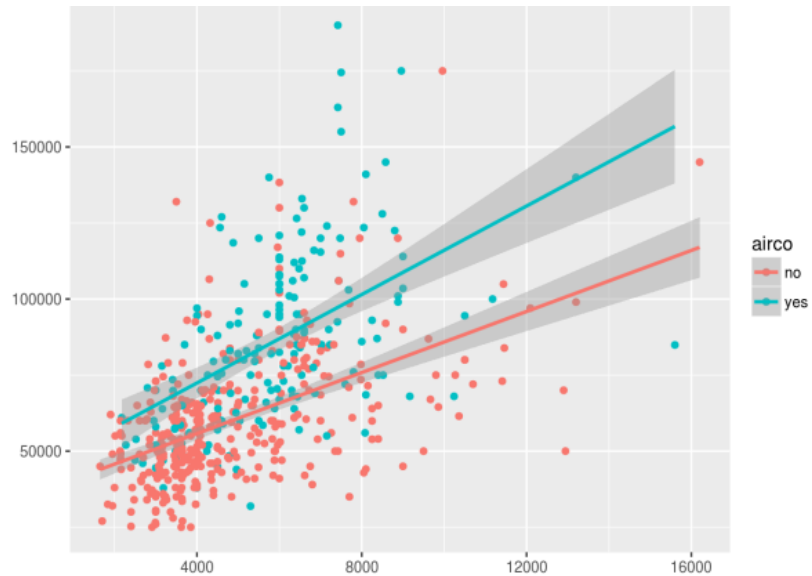
$$\{\mathbf{x} | \hat{f}_k(\mathbf{x}) = \hat{f}_\ell(\mathbf{x})\} = \{\mathbf{x} | (\hat{\theta}_{k,0} - \hat{\theta}_{\ell,0}) + (\hat{\boldsymbol{\theta}}_k - \hat{\boldsymbol{\theta}}_\ell)^T \mathbf{x} = 0\}$$

  is an hyperplane.

- The predicted output is :

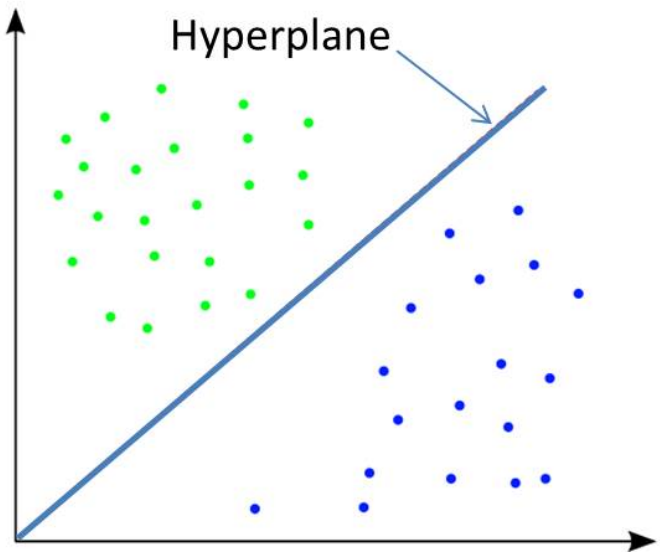
$$\hat{f}(\mathbf{x}) = \arg\max_k \hat{f}_k(\mathbf{x})$$

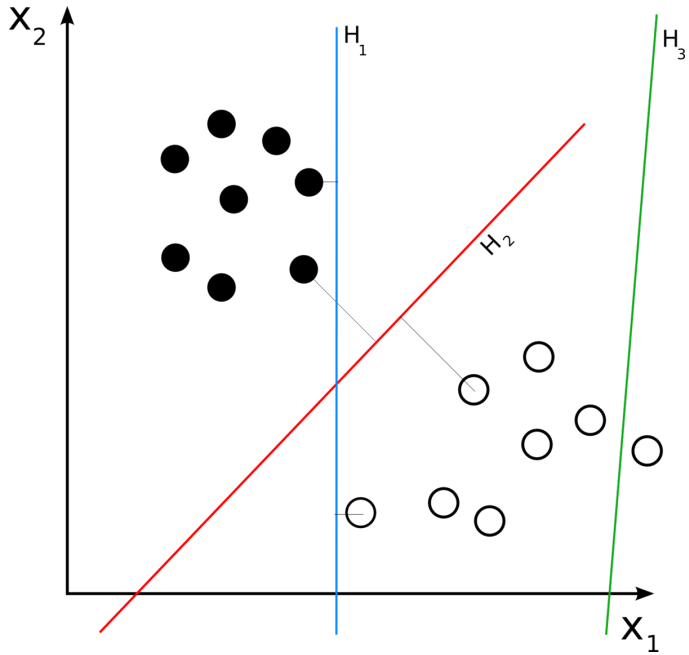
- This also works for transformations of $\hat{f}_k(\mathbf{x})$



How to obtain linear boundaries? (II)

- Directly looking for linear *separating hyperplanes*: support vector machines (SVMs)





Logistic regression

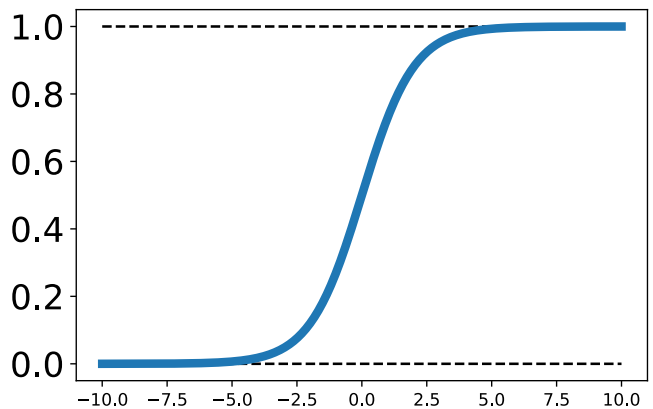
- It is a **classification method**: input $(\mathbf{x}_i)_i \in \mathbb{R}^d$ and $(y_i)_i \in \{+1, 0\}$.
- **Probabilistic model**: find a model $h_{\boldsymbol{\theta}}$ s.t. $\mathbb{P}(y = +1|\mathbf{x}) \approx h_{\boldsymbol{\theta}}(\mathbf{x})$.
- Bayes decision: $f(\mathbf{x}) = \text{argmax}_{k \in \{0,1\}} \mathbb{P}(y = k|\mathbf{x})$.

Logistic regression

- It is a **classification method**: input $(\mathbf{x}_i)_i \in \mathbb{R}^d$ and $(y_i)_i \in \{+1, 0\}$.
- **Probabilistic model**: find a model $h_{\boldsymbol{\theta}}$ s.t. $\mathbb{P}(y = +1|\mathbf{x}) \approx h_{\boldsymbol{\theta}}(\mathbf{x})$.
- Bayes decision: $f(\mathbf{x}) = \text{argmax}_{k \in \{0,1\}} \mathbb{P}(y = k|\mathbf{x})$.

The sigmoid function

$\sigma(z) = 1/(1 + \exp(-z))$.



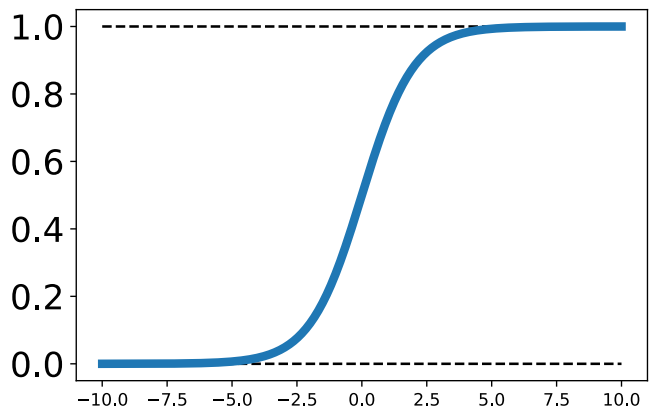
- Usually used to model probabilities.

Logistic regression

- It is a **classification method**: input $(\mathbf{x}_i)_i \in \mathbb{R}^d$ and $(y_i)_i \in \{+1, 0\}$.
- **Probabilistic model**: find a model $h_{\boldsymbol{\theta}}$ s.t. $\mathbb{P}(y = +1|\mathbf{x}) \approx h_{\boldsymbol{\theta}}(\mathbf{x})$.
- Bayes decision: $f(\mathbf{x}) = \text{argmax}_{k \in \{0,1\}} \mathbb{P}(y = k|\mathbf{x})$.

The sigmoid function

$\sigma(z) = 1/(1 + \exp(-z))$.

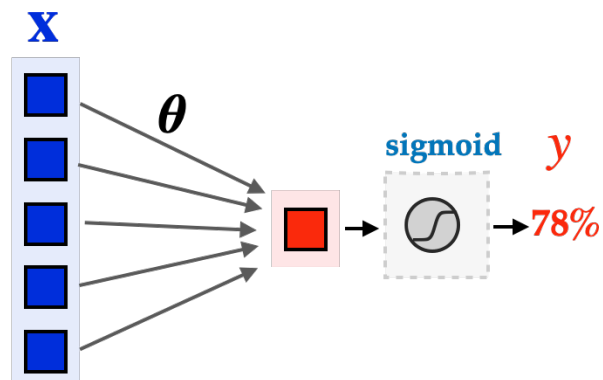


- Usually used to model probabilities.

The logistic regression model

The model is $\mathbb{P}(y = +1|\mathbf{x}) = \sigma(\boldsymbol{\theta}^\top \mathbf{x} + b)$.

- $\boldsymbol{\theta} \in \mathbb{R}^d$ are weights, $b \in \mathbb{R}$ is a bias that are to be optimized.
- It is a generalized linear model.
- Is is also a one layer neural-network (no hidden layer).



Logistic regression

One property

$$\mathbb{P}(y = 0|\mathbf{x}) = 1 - \mathbb{P}(y = 1|\mathbf{x}) = 1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x} + b) = \sigma(-(\boldsymbol{\theta}^\top \mathbf{x} + b))$$

Logistic regression

One property

$$\mathbb{P}(y = 0|\mathbf{x}) = 1 - \mathbb{P}(y = 1|\mathbf{x}) = 1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x} + b) = \sigma(-(\boldsymbol{\theta}^\top \mathbf{x} + b))$$

Maximum likelihood estimation

Find $\boldsymbol{\theta} \in \mathbb{R}^d, b \in \mathbb{R}$ that minizes the cross-entropy (board)

$$-\sum_i y_i \log \mathbb{P}(y_i = 1|\mathbf{x}_i) + (1 - y_i) \log \mathbb{P}(y_i = 0|\mathbf{x}_i)$$

Logistic regression

One property

$$\mathbb{P}(y = 0|\mathbf{x}) = 1 - \mathbb{P}(y = 1|\mathbf{x}) = 1 - \sigma(\boldsymbol{\theta}^\top \mathbf{x} + b) = \sigma(-(\boldsymbol{\theta}^\top \mathbf{x} + b))$$

Maximum likelihood estimation

Find $\boldsymbol{\theta} \in \mathbb{R}^d, b \in \mathbb{R}$ that minizes the cross-entropy (board)

$$-\sum_i y_i \log \mathbb{P}(y_i = 1|\mathbf{x}_i) + (1 - y_i) \log \mathbb{P}(y_i = 0|\mathbf{x}_i)$$

Minimizing the logistic loss

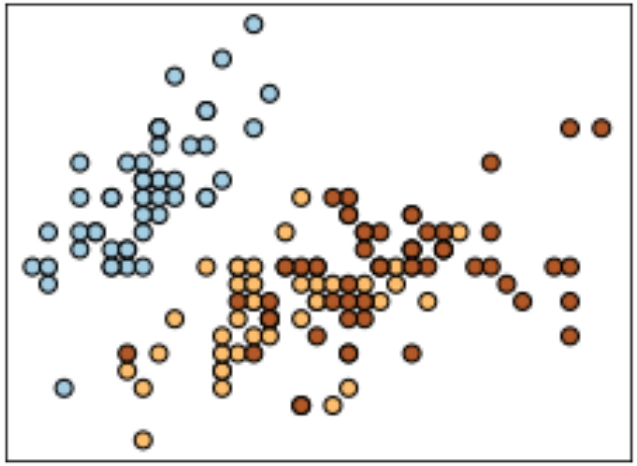
$$\min_{\boldsymbol{\theta}, b} \sum_{i=1}^{n} -y_i(\boldsymbol{\theta}^\top \mathbf{x}_i + b) + \log\left[1 + \exp\left(\boldsymbol{\theta}^\top \mathbf{x}_i + b\right)\right] .$$

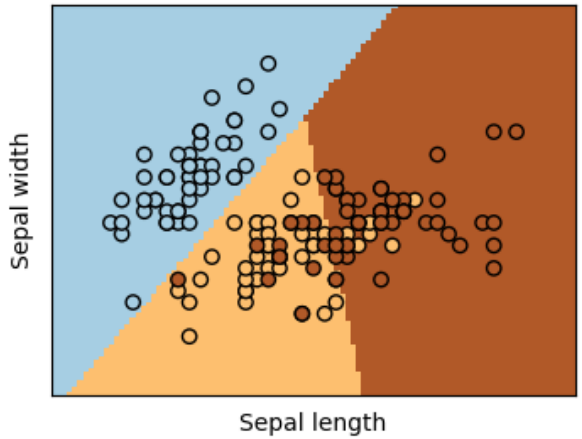
- **Convex problem**, can be solved with (Quasi) Newton's method.
- It is a linear decision boundary.
- Extends also to multi-class classification by modeling

$$\mathbb{P}(y = k|\mathbf{x}) = \text{softmax}((\boldsymbol{\theta}_k^\top \mathbf{x} + \mathbf{b}_k)_{k \in [\![K]\!]}) \tag{1}$$

and minimizing cross-entropy.

Logistic regression with Scikit-learn





Principle

- First split the data into train and test
- LogisticRegression() builds a logistic model
- logreg.fit fits the model to the data
- accuracy_score and confusion_matrix allow to evaluate the model

Python code

```
1  from sklearn.model_selection import
       train_test_split
2  from sklearn.linear_model import
       LogisticRegression
3  from sklearn import metrics
4
5  X_train, X_test, y_train, y_test =
       train_test_split(X, y)
6  logreg = LogisticRegression(C=1e5)
7  logreg.fit(X, Y)
8
9  pred= model2.predict(X_test)
10 print(metrics.accuracy_score(y_test, pred))
11 print(metrics.confusion_matrix(y_test, pred))
```

From logistic regression to LDA/QDA

A model

- In log-reg we model $\mathbb{P}(Y = k|X = x)$.
- In LDA/QDA we will instead model $\mathbb{P}(X = x|Y = k)$.

Bayes theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A|B)$ is called the *posterior*. Example: "probability of having cancer given that the person is a smoker".
- $P(B|A)$ is called the *likelihood*; Example: "probability of being a smoker given that the person has cancer".
- $P(A)$ is called the *prior*; this is the probability of our hypothesis without any additional prior information. Example: "probability of having cancer".
- $P(B)$ is called the *marginal likelihood*; this is the total probability of observing the evidence. Example: "probability of being a smoker".

Linear discriminant analysis

From Bayes theorem:

$$\mathbb{P}(Y = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{\ell=1}^{K} f_\ell(x)\pi_\ell}$$

with

- $f_k(x) = \mathbb{P}(X = x|Y = k)$ the conditional density of X in class k
- $\pi_k = \mathbb{P}(Y = k)$ the prior probability of class k, $\sum_{k=1}^{K} \pi_k = 1$

Many techniques use models for the class densities.

Example

- LDA and QDA use gaussian densities
- mixtures of Gaussians allow for nonlinear boundaries

Linear discriminant analysis (LDA)

We model each class density as multivariate Gaussian

$$f_k(x) = \frac{1}{\sqrt{(2\pi)^d det(\Sigma_k)}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)}.$$

LDA arises in the special case when $\Sigma_k = \Sigma$, $\forall k$.
If we compare two classes:

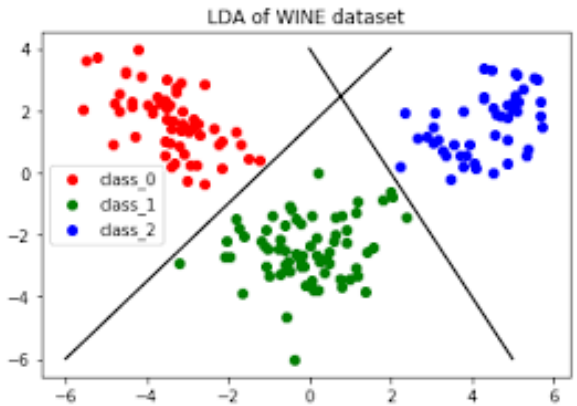
$$\log \frac{P(y=k|X=x)}{P(y=\ell|X=x)} =$$

$$\log \frac{f_k(x)}{f_\ell(x)} + \log \frac{\pi_k}{\pi_\ell} =$$

$$\log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k + \mu_\ell)^T \Sigma^{-1}(\mu_k - \mu_\ell)$$

$$+ x^T \Sigma^{-1}(\mu_k - \mu_\ell)$$

which is linear in $x \rightarrow$ the decision boundary between two classes is linear:



$$P(y=k|X=x) = P(y=\ell|X=x)$$
$$\rightarrow \log \frac{P(y=k|X=x)}{P(y=\ell|X=x)} = 0$$

Linear discriminant analysis (LDA)

The samples can be equivalently divided into classes using the linear discriminant function:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k \Sigma^{-1} \mu_k + \log \pi_k$$

and $\hat{y}(x) = \arg\max_k \delta_k(x)$.

The parameters of the gaussians distributions are not known, we need to estimate them from the training data:

- $\hat{\pi}_k = n_k / n$, with n_k the number of training samples in class k
- $\hat{\mu}_k = \sum_{y_i = k} x_i / n_k$
- $\hat{\Sigma} = \sum_{k=1}^{K} \sum_{y_i = k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (n - K)$

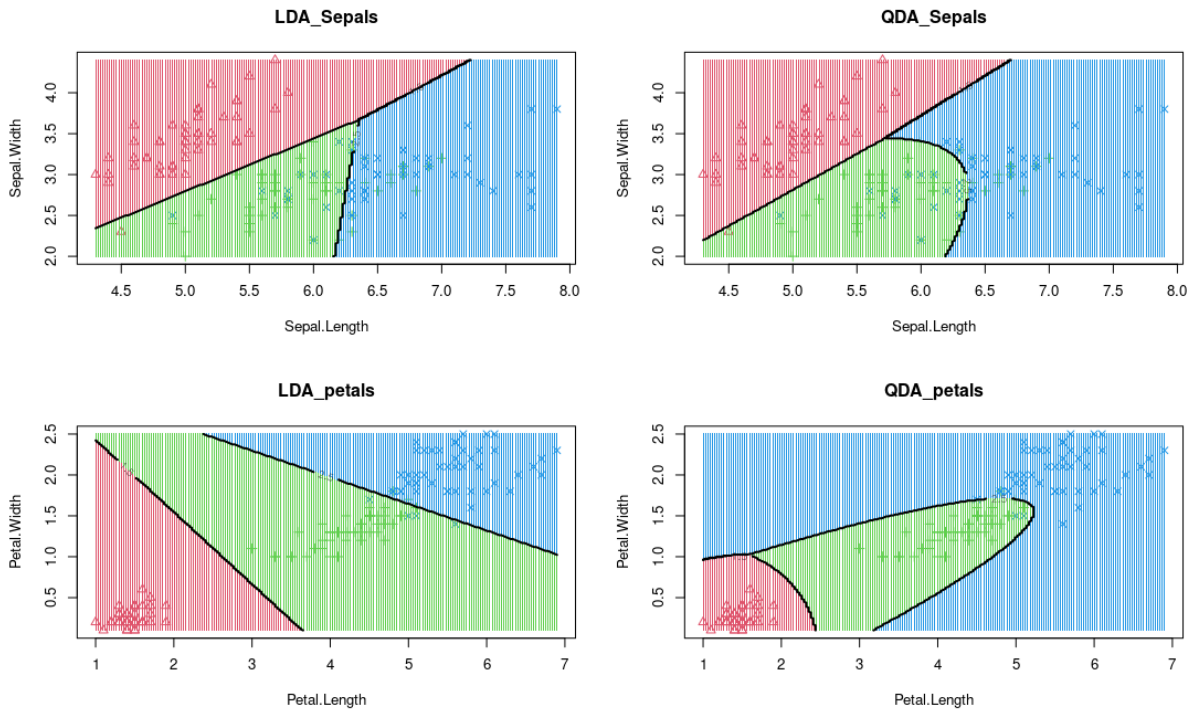
Quadratic discriminant analysis (QDA)

The samples are divided into classes using the quadratic discriminant function:

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k$$

and $\hat{y}(x) = \arg\max_k \delta_k(x)$.

The decision boundaries are quadratic: $\{x|\delta_k(x) = \delta_\ell(x)\}$



Evaluate classification models

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

- Accuracy = (TP+TN) / N → no good for imbalanced datasets
- Confusion matrix
  - *Precision* : the proportion of data correctly predicted as positive (percentage of good positives in the positive class)

$$Prec = (TP)/(TP + FP)$$

  - *Recall* (Sensitivity, True Positive Rate): how good your model is at correctly predicting positive cases. It's the proportion of positive cases that were correctly identified.

$$Recall = (TP)/(TP + FN)$$

Support vector machines: binary classification

Given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $y_i = -1$ or $y_i = 1$

Separating hyperplane

- We look for a hyperplane that separates samples in the training set belonging to different classes:
  - $\mathcal{P} = \{\mathbf{x} \in \mathbb{R}^d | y(x) = +1\}$
  - $\mathcal{N} = \{\mathbf{x} \in \mathbb{R}^d | y(x) = -1\}$

Hyperplane: for $\boldsymbol{\theta} \in \mathbb{R}^d, b \in \mathbb{R}$

$$H = H_{\boldsymbol{\theta},b} = \{\mathbf{x} \in \mathbb{R}^d \mid h(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} + b = 0 \ \}$$

- New samples are assigned to a class according to the sign of function h.

The margin

- The separating hyperplane is not unique: for each the margin $\rho(\boldsymbol{\theta}, b)$ is defined as:

$$\rho(\boldsymbol{\theta}, b) = \min \frac{|\boldsymbol{\theta}^T \mathbf{x} + b|}{\|\boldsymbol{\theta}\|}$$

- The margin is the distance of the closest sample to the hyperplane:

$$\rho(\boldsymbol{\theta}, b) = \min_{\mathbf{x}_i} d(\mathbf{x}_i, H)$$

Optimal hyperplane

- The optimal hyperplane is the one that maximizes the margin:

$$\max_{\boldsymbol{\theta} \in \mathbb{R}^d, b \in \mathbb{R}} \rho(\boldsymbol{\theta}, b)$$

- Minimizes the probability of errors for the classification of new samples

Linearly separable case

- If features are linearly separable it exists an hyperplane H such that: $h(\mathbf{x}_i) > 0$ for all $\mathbf{x}_i \in \mathcal{P}$ and $h(\mathbf{x}_i) < 0$ for all $\mathbf{x}_i \in \mathcal{N}$.
- There exist then $\boldsymbol{\theta} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that for $\epsilon > 0$:

$$\boldsymbol{\theta}^T \mathbf{x}_i + b \geq \epsilon, \text{ for all } \mathbf{x}_i \in \mathcal{P},$$
$$\boldsymbol{\theta}^T \mathbf{x}_i + b \leq -\epsilon, \text{ for all } \mathbf{x}_i \in \mathcal{N}.$$

- By scaling, WLOG, we can consider $\boldsymbol{\theta} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that :

$$\boldsymbol{\theta}^T \mathbf{x}_i + b \geq 1, \text{ for all } \mathbf{x}_i \in \mathcal{P},$$
$$\boldsymbol{\theta}^T \mathbf{x}_i + b \leq -1, \text{ for all } \mathbf{x}_i \in \mathcal{N}. \tag{2}$$

- We will define a separating hyperplane each $H = H_{\boldsymbol{\theta},b}$ with $(\boldsymbol{\theta}, b) \in \mathbb{R}^d \times \mathbb{R}$ satisfying (2).

Linearly separable case

Finding the optimal hyperplane requires solving

$$\max_{\boldsymbol{\theta}\in\mathbb{R}^n, b\in\mathbb{R}} \min_{\mathbf{x}_i\in\mathcal{P}\cup\mathcal{N}} \frac{|\boldsymbol{\theta}^T\mathbf{x}_i + b|}{\|\boldsymbol{\theta}\|}$$

We will prove that the optimal hyperplane exists and is unique, and that it can be found solving an equivalent problem:

$$\begin{aligned} &\min_{\boldsymbol{\theta}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2}\|\boldsymbol{\theta}\|^2 \\ \text{s.t. } &\boldsymbol{\theta}^T\mathbf{x}_i + b \geq 1, \text{ for all } \mathbf{x}_i \in \mathcal{P}, \\ &\boldsymbol{\theta}^T\mathbf{x}_i + b \leq -1, \text{ for all } \mathbf{x}_i \in \mathcal{N}. \end{aligned}$$

Theoretical results

Lemma 1

For all separating hyperplanes it holds

$$\rho(\boldsymbol{\theta}, b) \geq \frac{1}{\|\boldsymbol{\theta}\|}$$

(from the definition of ρ, because $|\boldsymbol{\theta}^T \mathbf{x}_i + b| \geq 1$ for all $\mathbf{x}_i \in \mathcal{P} \cup \mathcal{N}$.)

Lemma 2

For all separating hyperplane $(\boldsymbol{\theta}, b)$ it exists another separating hyperplane $(\bar{\boldsymbol{\theta}}, \bar{b})$ such that

$$\rho(\boldsymbol{\theta}, b) \leq \rho(\bar{\boldsymbol{\theta}}, \bar{b}) = \frac{1}{\|\bar{\boldsymbol{\theta}}\|}.$$

Moreover, there exist (at least) two points $\mathbf{x}^+$ and $\mathbf{x}^-$ such that

$$\bar{\boldsymbol{\theta}}^T \mathbf{x}^+ + \bar{b} = 1$$
$$\bar{\boldsymbol{\theta}}^T \mathbf{x}^- + \bar{b} = -1$$

Theoretical results

Theorem 1

The problem

$$\begin{aligned} &\min_{\boldsymbol{\theta}\in\mathbb{R}^d, b\in\mathbb{R}} \|\boldsymbol{\theta}\|^2 \\ \text{s.t. } &\boldsymbol{\theta}^T\mathbf{x}_i + b \geq 1, \text{ for all } \mathbf{x}_i \in \mathcal{P}, \\ &\boldsymbol{\theta}^T\mathbf{x}_i + b \leq -1, \text{ for all } \mathbf{x}_i \in \mathcal{N}. \end{aligned} \tag{3}$$

admits a unique solution $(\boldsymbol{\theta}^*, b^*)$.

Notice that this is a convex quadratic programming problem.

Theoretical results

Theorem 2

If $(\boldsymbol{\theta}^*, b^*)$ is the solution of problem (3), it is also the only solution of

$$\begin{aligned} &\max_{\boldsymbol{\theta}\in\mathbb{R}^d, b\in\mathbb{R}} \rho(\boldsymbol{\theta}, b) \\ \text{s.t. } &\boldsymbol{\theta}^T \mathbf{x}_i + b \geq 1, \text{ for all } \mathbf{x}_i \in \mathcal{P}, \\ &\boldsymbol{\theta}^T \mathbf{x}_i + b \leq -1, \text{ for all } \mathbf{x}_i \in \mathcal{N}. \end{aligned}$$

Proof

From Lemma 1 and 2 for every separating hyperplane, we have:

$$\frac{1}{\|\boldsymbol{\theta}\|} \leq \rho(\boldsymbol{\theta}, b) \leq \rho(\bar{\boldsymbol{\theta}}, \bar{b}) = \frac{1}{\|\bar{\boldsymbol{\theta}}\|} \leq \frac{1}{\|\boldsymbol{\theta}^*\|}$$

So for $(\boldsymbol{\theta}^*, b^*)$ we get $\rho(\boldsymbol{\theta}^*, b^*) = \frac{1}{\|\boldsymbol{\theta}^*\|}$ so it is the optimal hyperplane.

How to solve (3)?

Usually the dual problem is solved:

$$\min_{\lambda \in \mathbb{R}^m} \frac{1}{2} \boldsymbol{\lambda}^T Z^T Z \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda}$$
$$\text{s.t. } \boldsymbol{\lambda}^T \mathbf{y} = 0, \ \boldsymbol{\lambda} \geq 0$$

with

$$Z = [y_1 \mathbf{x}_1, \ldots, y_m \mathbf{x}_m], \quad \mathbf{e} = [1, \ldots, 1]^T$$

and m number of training samples. Solving this we find $\boldsymbol{\lambda}^*$ and we can recover $\boldsymbol{\theta}^*$ and b^* as :

$$\boldsymbol{\theta}^* = \sum_{i=1}^{m} \lambda_i^* y_i \mathbf{x}_i \rightarrow \text{support vectors}$$
$$\lambda_i [y_i ((\boldsymbol{\theta}^*)^T \mathbf{x}_i + b^*) - 1] = 0, i = 1, \ldots, m$$

The decision function is then:

$$f(\mathbf{x}) = \text{sgn}((\boldsymbol{\theta}^*)^T \mathbf{x} + b^*)$$

Non separable case

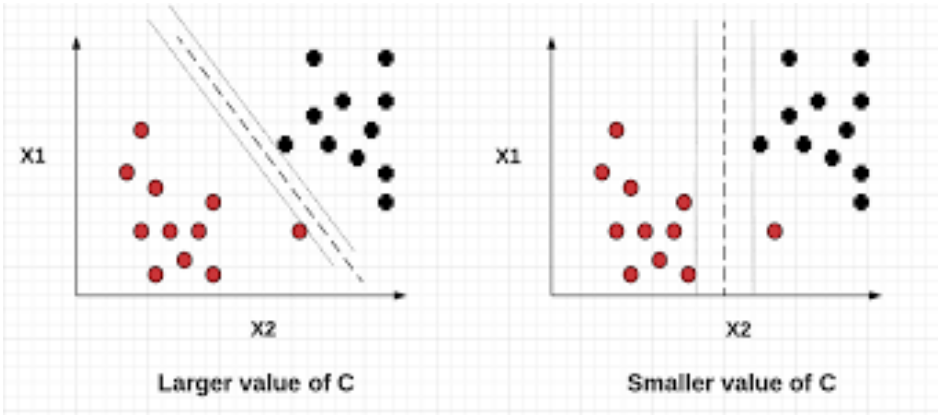
If the features are not linearly separable it is necessary to allow the presence of some outliers inserting some slack variables $\zeta_i \geq 0$ $i = 1, \ldots, m$:

$$\boldsymbol{\theta}^T \mathbf{x}_i + b \geq 1 - \zeta_i \text{ for all } \mathbf{x}_i \in \mathcal{P},$$
$$\boldsymbol{\theta}^T x_i + b \leq -1 + \zeta_i \text{ for all } \mathbf{x}_i \in \mathcal{N}.$$

- If $\mathbf{x}_i$ is incorrectly classified $\zeta_i > 1$, so $\sum_{i=1}^{m} \zeta_i$ is an upper bound of the number of training features misinterpreted:

$$\min_{\omega, b, \zeta} \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C \sum_{i=1}^{m} \zeta_i$$
$$\text{s.t. } y_i(\boldsymbol{\theta}^T \mathbf{x}_i + b) \leq 1 - \zeta_i,$$
$$\zeta_i \geq 0, \ i = 1, \ldots, m.$$

The value of C



Which problem to solve in practice?

Usually the dual problem is solved:

$$\min_{\lambda \in \mathbb{R}^m} \frac{1}{2} \boldsymbol{\lambda}^T Z^T Z \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda}$$
$$\text{s.t. } \boldsymbol{\lambda}^T \mathbf{y} = 0, \; 0 \leq \boldsymbol{\lambda} \leq C$$

with

$$Z = [y_1 \mathbf{x}_1, \ldots, y_m \mathbf{x}_m], \quad \mathbf{e} = [1, \ldots, 1]^T$$

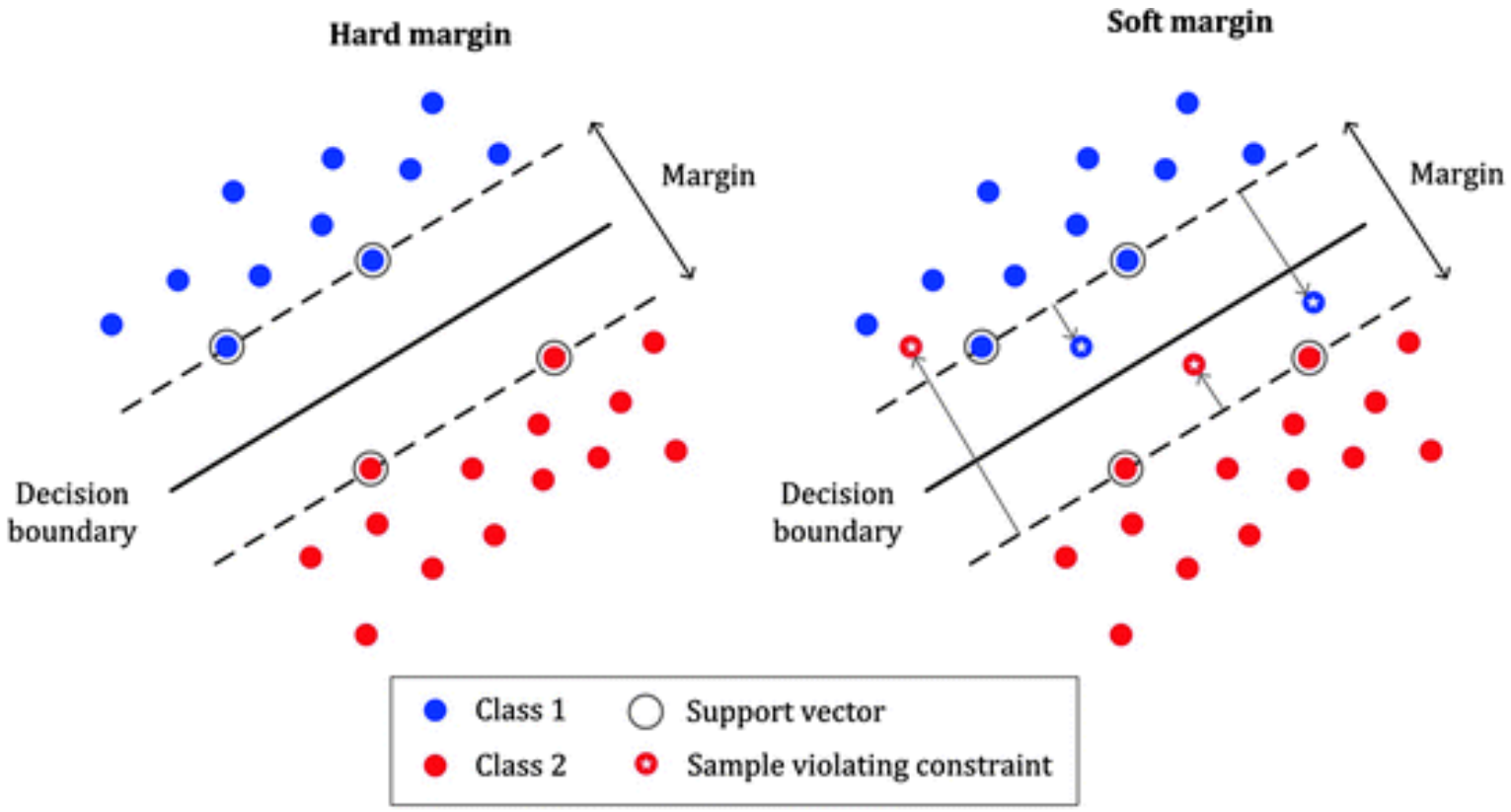
and m number of training samples. Solving this we find $\boldsymbol{\lambda}^*$ and we can recover $\boldsymbol{\theta}^*$ and b^* as :

$$\boldsymbol{\theta}^* = \sum_{i=1}^{m} \lambda_i^* y_i \mathbf{x}_i \rightarrow \text{support vectors}$$
$$\lambda_i [y_i ((\boldsymbol{\theta}^*)^T \mathbf{x}_i + b^*) - 1] = 0, i = 1, \ldots, m$$

The decision function is then:

$$f(\mathbf{x}) = \text{sgn}((\boldsymbol{\theta}^*)^T \mathbf{x} + b^*)$$

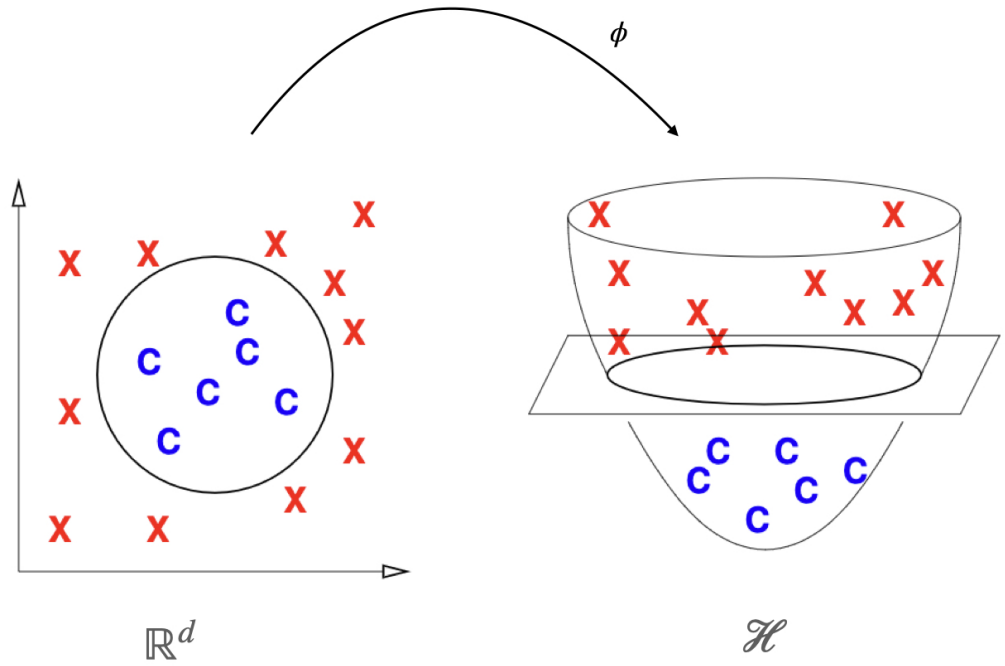
Hard margin SVM vs soft margin SVM



Nonlinear SVM

The kernel trick

Move to an higher dimensional space to make the features separable



Kernel function

Given $\Omega \subset \mathbb{R}^d$, $k : \Omega \times \Omega \to \mathbb{R}$ is a kernel function if $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \Omega$, where $\phi : \Omega \to \mathcal{H}$ with $\mathcal{H}$ an Hilbert space.

Linear SVM in the feature space

Idea

Choose a *feature space* $\mathcal{H}$ of dimension higher than d. Apply the method previously seen to $\phi(\mathbf{x}_i)$ and look for a separating hyperplane in $\mathcal{H}$.

$$\min_{\lambda \in \mathbb{R}^m} \frac{1}{2} \boldsymbol{\lambda}^T Z^T Z \boldsymbol{\lambda} - \mathbf{e}^T \boldsymbol{\lambda}$$
$$\text{s.t. } \boldsymbol{\lambda}^T \mathbf{y} = 0,\ 0 \leq \boldsymbol{\lambda} \leq C$$

with

$$Z = [y_1 \phi(\mathbf{x}_1), \ldots, y_m \phi(\mathbf{x})_m], \quad \mathbf{e} = [1, \ldots, 1]^T$$

The decision function is then:

$$f(\mathbf{x}) = \text{sgn}((\boldsymbol{\theta}^*)^T \phi(\mathbf{x}) + b^*) = \text{sgn}(\sum_{i=1}^{m} \lambda_i^* y_i k(\mathbf{x}_i, \mathbf{x}) + b^*)$$

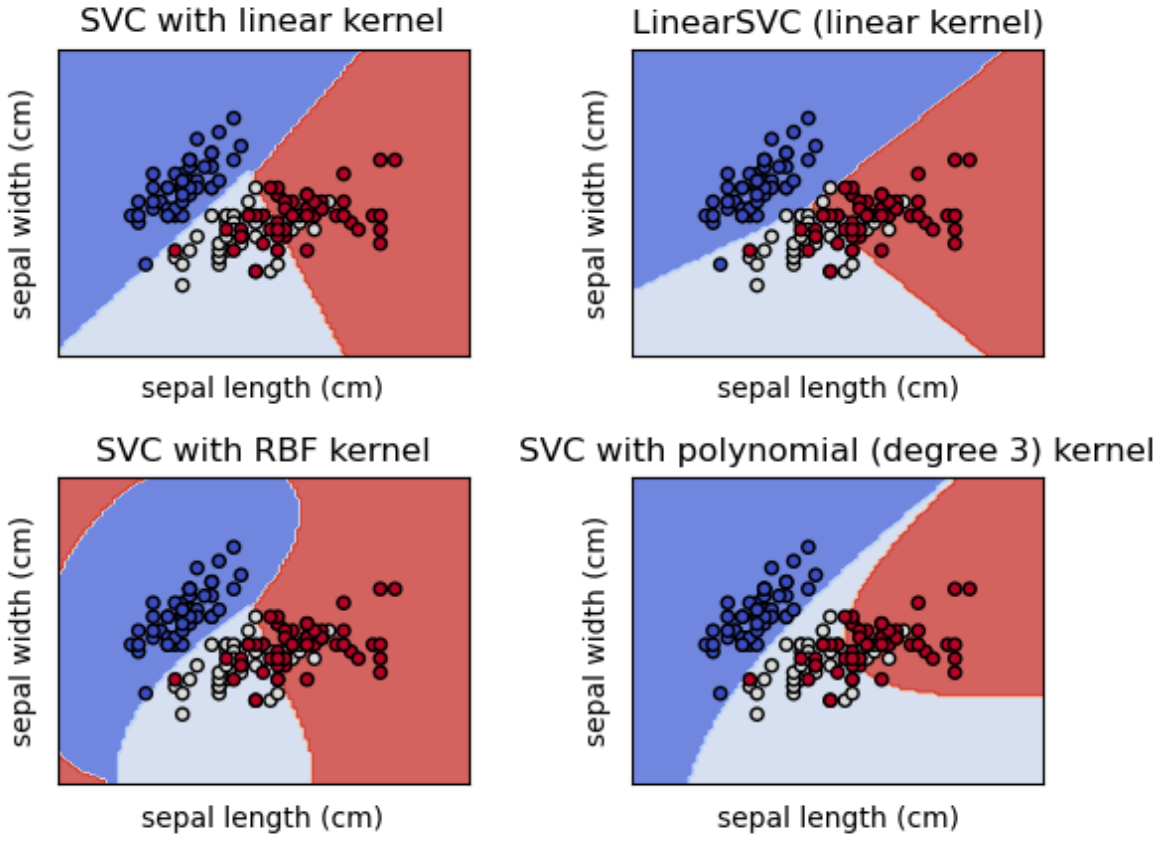
f is linear in the *feature space*, nonlinear in the *input space*

Most common kernels

- Polynomial kernel: $k(x, z) = (x^T z + 1)^p$, $p \in \mathbb{N}$, $p \geq 1$
- Gaussian kernel (or RBF) $k(x, z) = e^{-\|x-z\|^2/2\sigma^2}$
- Hyperbolic tangent $k(x, z) = \tanh(\beta x^T z + \gamma)$, $\beta, \gamma \in \mathbb{R}$

Libraries for SVM

- LIBSVM: `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`, LIBLINEAR: `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`
- sklearn: sklearn.svm.SVC(C=1.0, kernel='rbf'..) (from LIBSVM)
- sklearn: sklearn.svm.LinearSVC(penalty='l2', ..) (from LIBLINEAR)



References I

- The elements of statistical learning, Hastie, Tibshirani, Friedman, Sprienger (2009)
- Statistical learning theory, Vapnik, Wiley, New York (1998)
- The nature of statistical learning theory, Vapnik, Springer-Verlag, New York (1995)