(x_i, y_i) – data $(i = 1, \dots, n)$ $\quad x_i \in \mathbb{R}^p$, $y_i \in \{-1, 1\}$

support vector classifier
maximal margin classifier – fix $C \geq 0$ (for $C = 0$: maximal margin classifier)

$$\text{maximise}_{\beta_0 \in \mathbb{R},\, \beta_1 \in \mathbb{R}^p,\, \varepsilon_i,\, M} \quad M$$

subject to $|\beta_1|^2 = 1$

and $y_i \cdot (\beta_0 + \beta_1 \cdot x_i) \geq M(1 - \varepsilon_i)$, $i = 1, 2, \dots, n$

$\varepsilon_i \geq 0$, $\sum_{i=1}^{n} \varepsilon_i \leq C$

$\beta_0 + \beta_1 x = 0$

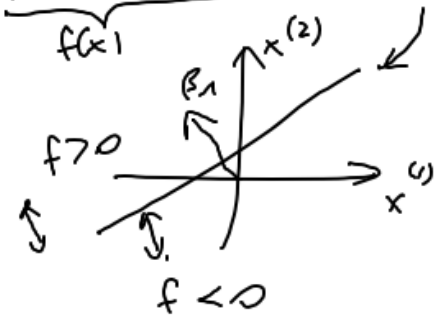
for example: $p = 2$

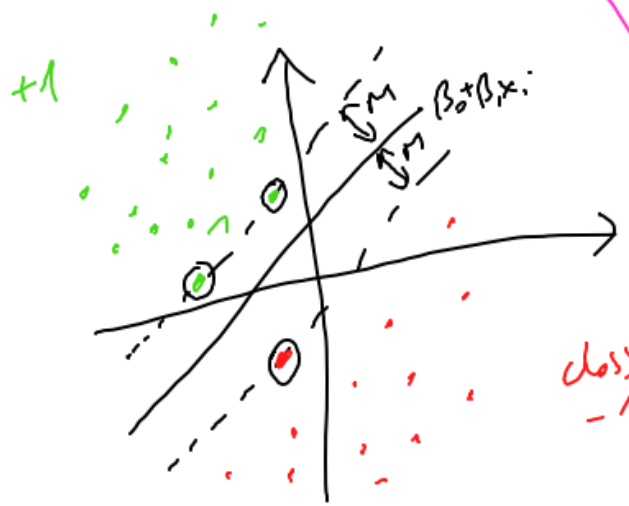
$\beta_1 = (\beta_1^{(1)}, \beta_1^{(2)})$

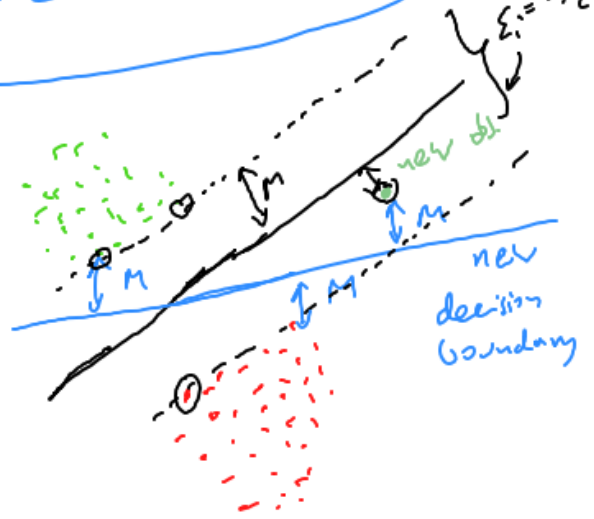
$x = (x^{(1)}, x^{(2)})$

$\beta_0 + \beta_1^{(1)} x^{(1)} + \beta_1^{(2)} x^{(2)} = 0$

$f(x)$







+ distance $(x_i,$ hyperplane$)$, if x_i is on the right side of the hyperplane

− distance $(x_i,$ −"− $)$, −"− (wrong side)

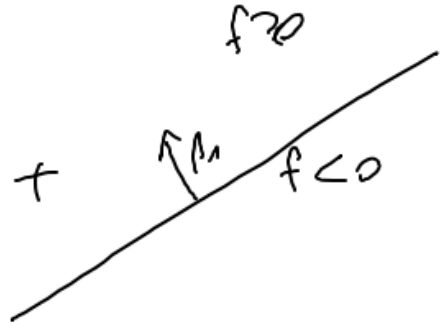
if $|\beta_1| = 1$, then $f(x_0)$ is the signed distance of x_0 to the hyperplane $\{x : \beta_0 + \beta_1 x = 0\}$

Hyperparameter C controls the bias-variance trade-off

The larger the constant C, the more observations influence the final form of the SVC.

The classification of $x \in \mathbb{R}^p$ is done by $\operatorname{sgn}\big(\underbrace{\beta_0 + \beta_1 x}_{f(x)}\big)$

The decision boundary is linear.



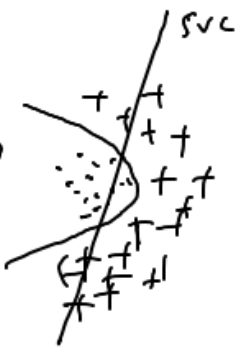
It turns out that to find the function $f(x) = \beta_0 + \beta_1 x$ one needs to know only ~~$x_i x_j$~~ $(i, j = 1, \ldots, n)$

$$x \cdot x_i = \langle x, x_i \rangle$$

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i \langle x, x_i \rangle + \beta_0$$

It turns out that one can replace the scalar products $\langle x, x_i \rangle$ by some other function $K(x, x_i)$ in the procedure of finding f, and in that way we may obtain a function f that classifies observations by $\operatorname{sgn} f(x)$. K cannot be completely arbitrary, e.g.

$$K(x, x') = (1 + \langle x, x' \rangle)^d$$
$$= \exp(-\gamma \|x - x'\|^2)$$
$$\ldots$$



machine
↓
SVM
↓
these produce some classifiers with nonlinear decision boundaries

SVC, SVM are (originally) binary classifier. We can make a multiclass classifier from binary classifiers in the following way:

- one vs rest (OvR): for each class $k=1,2,\dots,K$, construct a binary classifier for each of the setups:

  $\{1\}$ vs $\{2,\dots,K\}$

  $\{2\}$ vs $\{1,3,\dots,K\}$

  ⋮

  $\{K\}$ vs $\{1,2,\dots,K-1\}$

  binary K classifiers

  Then we may use the classification given by the most "confident" classifier among those that classify the sample as being in the single class

- one vs one (OvO): for each pair $j,k=1,\dots,K$, $j \neq k$ we construct a binary classifier for the setup

  $\{j\}$ vs $\{k\}$

  $\frac{K(K-1)}{2}$ classifiers

we may use majority vote (possibly with weight given by the confidence)

commonly used for SVC, SVM

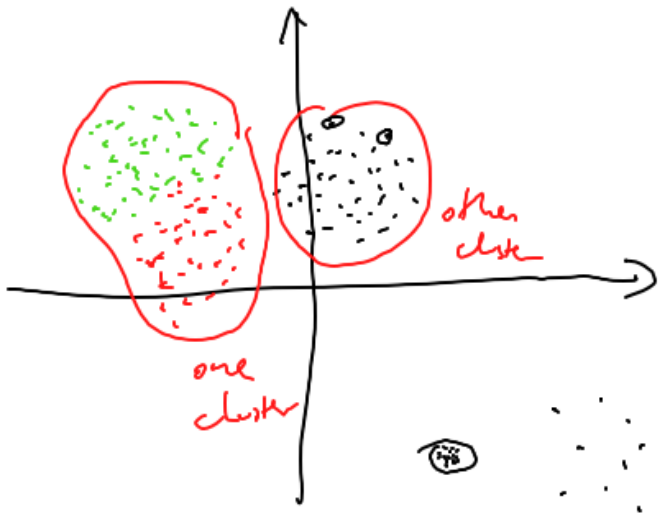
Clustering (unsupervised ML)

Goal: partition the data into distinct groups so that the observations within each group are quite similar to each other, while observations in different groups are quite different from each other

K-means clustering

$x_1, x_2, \dots, x_n \in \mathbb{R}^p$ – observations

- we specify the desired number of clusters K



Notation:

$C_1, C_2, \dots, C_K \subset \{1, 2, \dots, n\}$ subset of indices

$C_1 \cup C_2 \cup \dots \cup C_K = \{1, 2, \dots, n\}$

$C_i \cap C_j = \emptyset$ for $i \neq j$

then we call $C_1, \dots, C_K$ – clusters of (indices of) observations

Idea: clustering is good if the total within-cluster variation is as small as possible,

i.e. we minimize $\sum_{j=1}^{K} W(C_j)$, where $W(C_j)$ = within-cluster variation for the cluster C_j

common choice: $W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \|x_i - x_{i'}\|^2$

$\|x_i - x_{i'}\|^2 = \sum_{j=1}^{p} (x_i^{(j)} - x_{i'}^{(j)})^2$

Solving this minimisation problem is difficult, K-means algorithm gives a 'local' minimum.

K-means algorithm:

1) assign randomly each observation to some cluster $1, 2, \ldots, K$
(in such a way that each $C_j \neq \emptyset$)
This gives us the initial clusters.

2) Iterate until cluster assignments do not change anymore:

(a) for each cluster, compute its centroid: $\bar{x}_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$

(b) assign each observation to the cluster whose centroid is the closest
(w/r to Euclidean distance)

$\bar{x}_1$

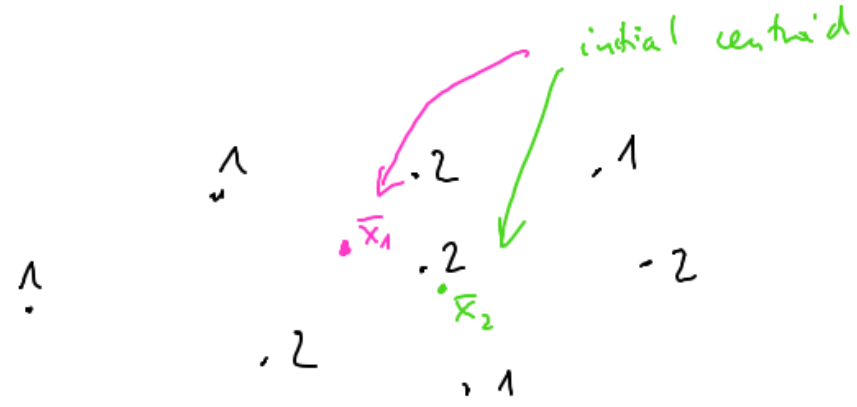
$\bar{x}_2$

One can prove that in each step 2a-2b the number $\sum_{j=1}^{K} W(C_j)$ decreases

In fact step 1) gives us only reasonable "centroids" $\bar{x}_k$ and we can replace it by some other choice of $\bar{x}_k$

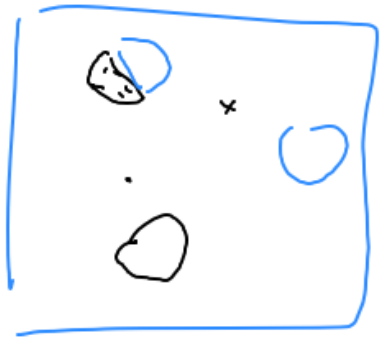
1. step

2a.



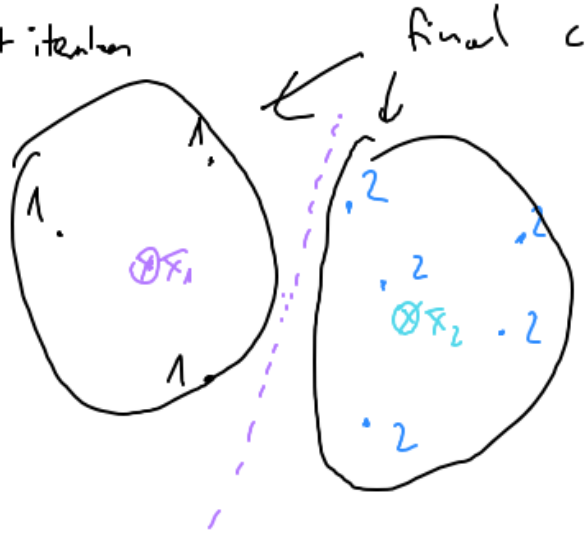
2b

and 2a - next iteration



2b - next iteration

final clustering



we stop.