Logistic Regression for Multiple Classes

Bernoulli and multinomial distributions

- The mulitnomial distribution of N draws from K categories with parameter vector $(\theta_1, \dots, \theta_K)$ (where $\sum_{k \leq K} \theta_k = 1$) has probabililty mass function

$$P(m_1, \dots, m_K | \theta_1, \dots, \theta_K) = \frac{N!}{m_1! \cdots m_K!} \prod_{k=1}^{K} \theta_k^{m_k} \qquad \text{where } m_k = \text{\# draws in category } k$$

- Note that $\text{Bernoulli}(p) = \text{Multinomial}(p, 1-p; N=1)$.

Logistic regression

- Recall two-class logistic regression is defined by $P(Y|\mathbf{x}) = \text{Bernoulli}(\sigma(\langle \mathbf{v}, \mathbf{x} \rangle - c))$.
- Idea: To generalize logistic regression to K classes, choose a separate weight vector $\mathbf{v}_k$ and offset c_k for each class k, and define $P(Y|\mathbf{x})$ by

$$\text{Multinomial}\big(\tilde{\sigma}(\langle \mathbf{v}_1, \mathbf{x} \rangle - c_1), \dots, \tilde{\sigma}(\langle \mathbf{v}_K, \mathbf{x} \rangle - c_K)\big)$$

where $\tilde{\sigma}(\langle \mathbf{v}_k, \mathbf{x} \rangle - c_k) = \frac{\sigma(\langle \mathbf{v}_k, \mathbf{x} \rangle - c_k)}{\sum_{j=1}^{K} \sigma(\langle \mathbf{v}_j, \mathbf{x} \rangle - c_j)}$. This definition ensures the $\tilde{\sigma}$-values add up to 1 over all classes.

Not examinable.

Logistic Regression for Multiple Classes

Logistic regression for K classes

The label y now takes values in $\{1, \dots, K\}$.

$$P(y|\mathbf{x}) = \prod_{k=1}^{K} \tilde{\sigma}(\langle \mathbf{v}_k, \mathbf{x} \rangle - c_k)^{\mathbb{I}\{y=k\}}$$

The negative log-likelihood becomes

$$L(\mathbf{v}_1, c_1, \dots, \mathbf{v}_K, c_K) \quad = \quad - \sum_{i \leq n,\, k \leq K} \mathbb{I}\{y = k\} \log \tilde{\sigma}(\langle \mathbf{v}_k, \tilde{\mathbf{x}}_i \rangle - c_k)$$

This can again be optimized numerically.

Comparison to two-class case

- Recall that $1 - \sigma(x) = \sigma(-x)$, and
  Bernoulli(p) = Multinomial$(p, 1-p)$ (with $N = 1$ draws)
- That means
$$\text{Bernoulli}\big(\sigma(\langle \mathbf{v}, \mathbf{x} \rangle - c)\big) \equiv \text{Multinomial}\big(\sigma(\langle \mathbf{v}, \mathbf{x} \rangle - c, \sigma(\langle -\mathbf{v}, \mathbf{x} \rangle + c)\big)$$
- That is: Two-class logistic regression as above is equivalent to multiclass logistic regression with $K = 2$ *provided* we choose $\mathbf{w}_2 = -\mathbf{w}_1$.

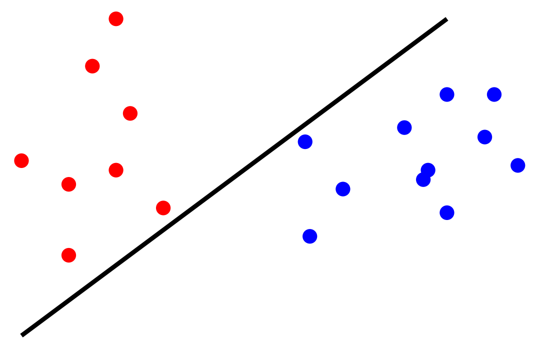
Not examinable.

Maximum Margin Classifiers

Maximum margin idea

Setting

Linear classification, two linearly separable classes.



Recall Perceptron

- Selects *some* hyperplane between the two classes.
- Choice depends on initialization, step size etc.

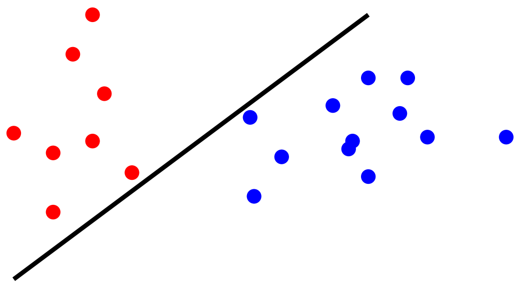
Maximum margin idea

To achieve good generalization (low prediction error), place the hyperplane "in the middle" between the two classes.

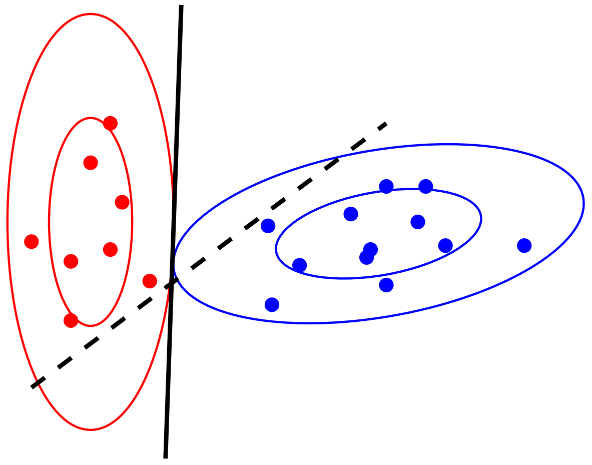
More precisely

Choose plane such that distance to closest point in each class is maximal. This distance is called the *margin*.

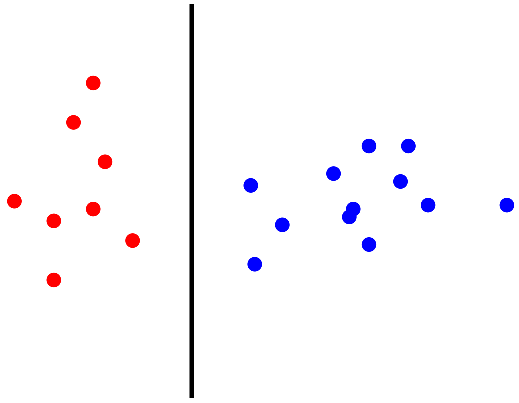
Generalization Error



Possible Perceptron solution



Good generalization under a specific distribution (here: Gaussian)



Maximum margin solution

Example: Gaussian data

- The ellipses represent lines of constant standard deviation (1 and 2 STD respectively).
- The 1 STD ellipse contains $\sim 68.3\%$ of the probability mass ($\sim 95.5\%$ for 2 STD; $\sim 99.7\%$ for 3 STD).

Optimal generalization: Classifier should cut off as little probability mass as possible from either distribution.

Without distributional assumption: Max-margin classifier

- Philosophy: Without distribution assumptions, best guess is symmetric.
- In the Gaussian example, the max-margin solution would *not* be optimal.

NEXT: TWO TOOLS

Convex sets

- There is an inherent relationship between linear classification and a geometric property of shapes called *convexity*.
- Convex shapes have very useful properties, and we can use those for classification.

Constrained optimization

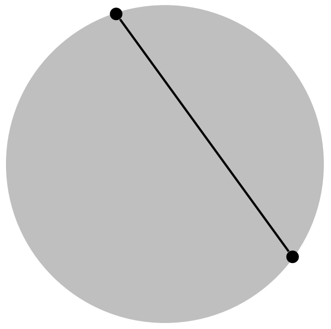
- The optimization problems we have considered before asked: What is the value of x for which $f(x)$ is as small as possible?
- A *constrained* optimization problem asks: Among all x which satisfy the property, which value makes $f(x)$ as small as possible?
- We use that to formulate the maximum margin problem as: Among all classifiers that separate the two classes, which one makes the margin as large as possible?

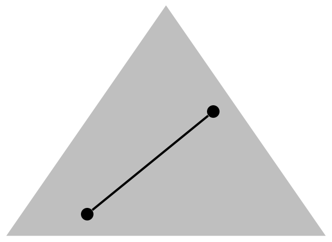
Convex sets

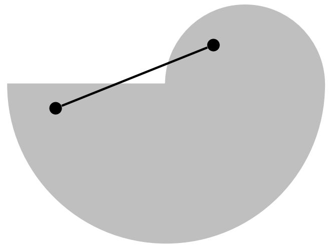
Definition

A set $A \subset \mathbb{R}^d$ is called **convex** if, for every two points $\mathbf{x}, \mathbf{y} \in A$, the straight line connecting $\mathbf{x}$ and $\mathbf{y}$ is completely contained in A.

Examples


convex


convex


not convex

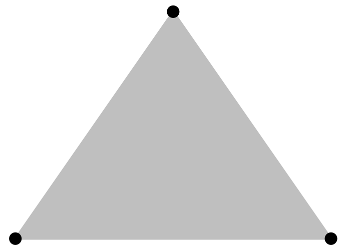
Extreme Points

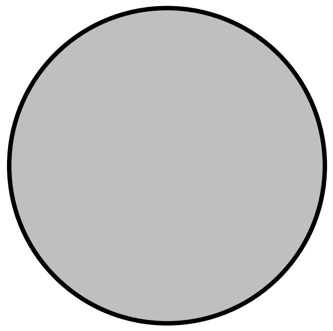
Extreme points

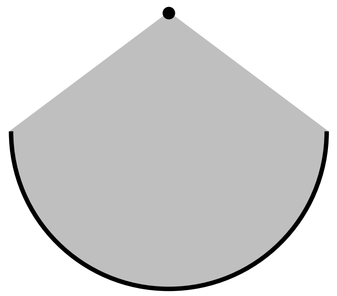
Let A be a convex set and $x \in A$. If x can be removed from A and $A \setminus \{x\}$ is still convex, then x is called an **extreme point** of A.

Examples

Extreme points are marked black.


finitely many extreme points


infinitely many extreme points


removing a point from the straight part of the boundary would leave a "hole", and the set would not be convex anymore.

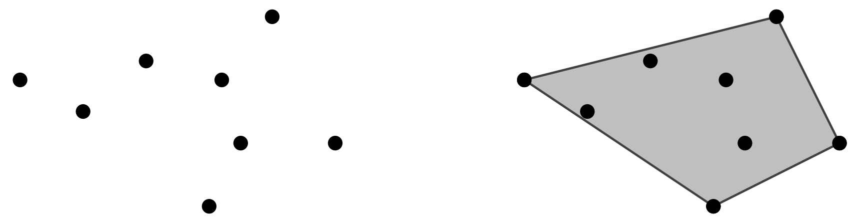
Informally

- If all segments of the boundary are straight lines or planes, the extreme points are exactly the "corner points" of the set.

Convex Hulls

Definition

If C is a finite set of points in $\mathbb{R}^d$, the **convex hull** $\text{conv}(C)$ of C is the smallest convex set that contains all points in C.



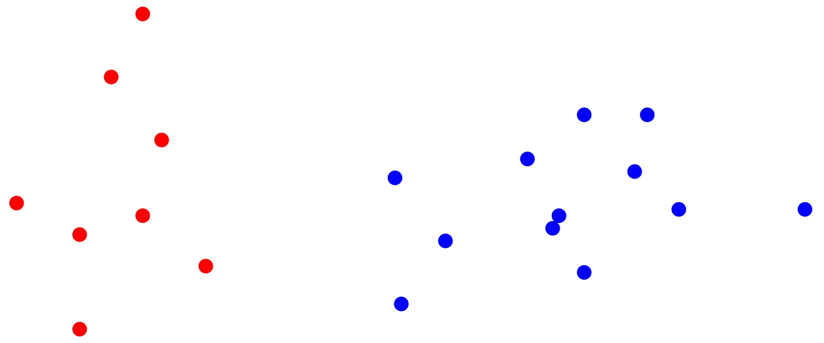
Note

- Each extreme point of $\text{conv}(C)$ is a point in the original set C.
- The convex hull is uniquely determined by C. (Every other convex in $\mathbb{R}^d$ either contains $\text{conv}(C)$, or does not contain all points in C.)
- Think of the convex hull as the shape we get by connecting the "outer" point of C.
- The importance of the convex hull for classification is that it defines which points in each training class are "outer" points (namely those which are extreme points of the convex hull).

Linear Classification and Convexity

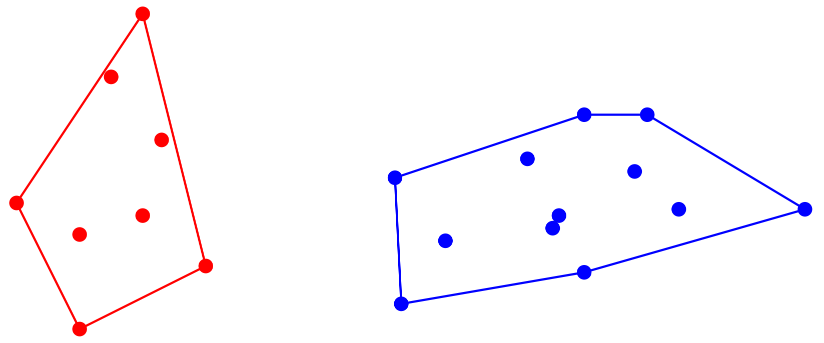
Observation

Where a separating affine plane may be placed depends on the "outer" points of the sets. Points in the center do not matter.



In geometric terms

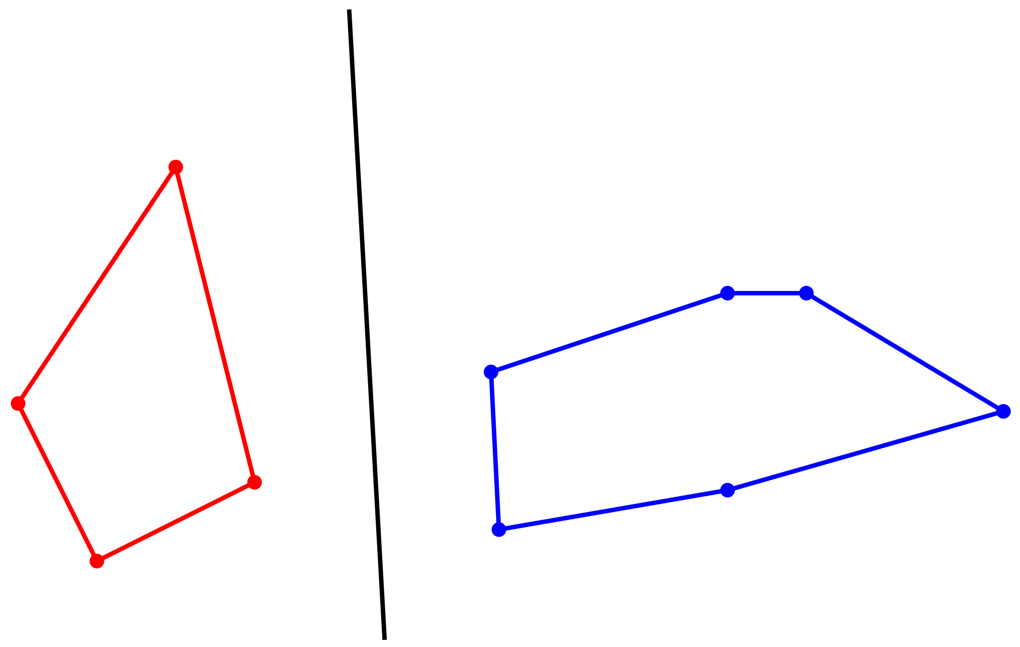
Substitute each class by its convex hull:



Convex Hulls and Classification

Key idea

There is an inherent relationship between convexity and linear classification: An affine plane separates two classes if and only if it separates their convex hulls.



Next

We have to formalize what it means for a hyperplane to be "in the middle" between two classes.

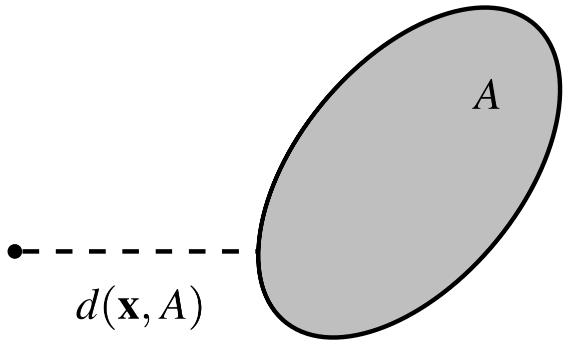
Distances to sets

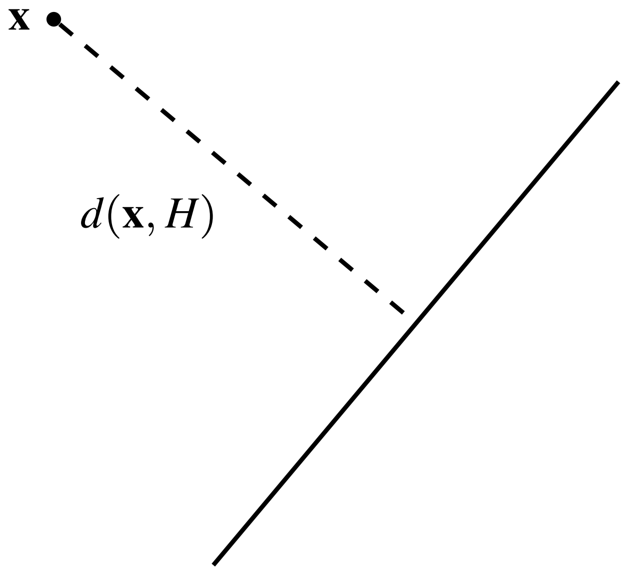
Definition

The **distance** between a point **x** and a set A the Euclidean distance between x and the closest point in A:

$$d(\mathbf{x}, A) := \min_{\mathbf{y} \in A} \|\mathbf{x} - \mathbf{y}\|$$

In particular, if $A = H$ is a hyperplane, $d(\mathbf{x}, H) := \min_{\mathbf{y} \in H} \|\mathbf{x} - \mathbf{y}\|$.



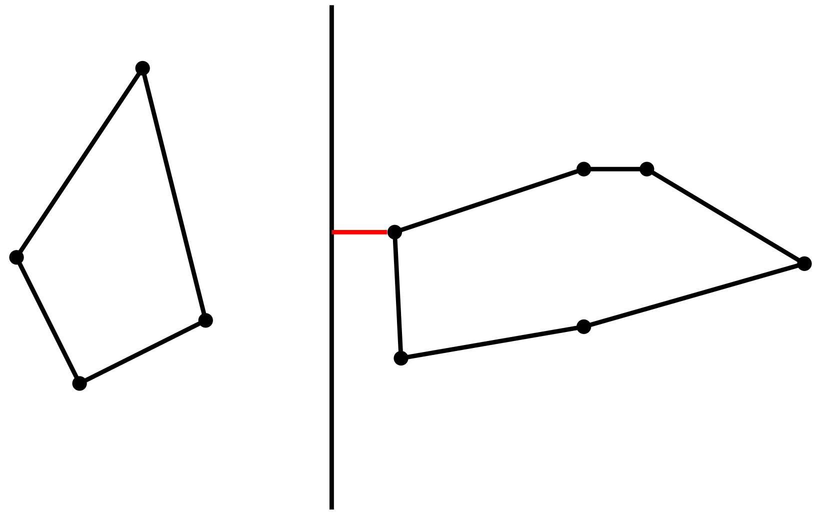


MARGIN

The **margin** of a classifier hyperplane H given two training classes is the shortest distance between the plane and any point in either set:

$$\text{margin} = \min_{x \in \text{ training data}} d(x, H)$$

Equivalently: The shortest distance to either of the convex hulls.



Idea in the following: H is "in the middle" when margin maximal.

Maximum Margin Problem

Basic problem

We want to find the affine plane $H(\mathbf{v}, c)$ that maximizes the distance to both data sets:

$$\text{maximize } d(H(\mathbf{v}, c), \text{training data}) \text{ over all } \mathbf{v}, c$$

Problem: The optimization algorithm can just move the plane further and further away from the data. We have to make sure H is "between the classes".

Maximum margin optimization problem

The problem we actually solve is

$$\begin{aligned} \text{maximize} \quad & d(H(\mathbf{v}, c), \text{training data}) \text{ over all } \mathbf{v}, c \\ \text{such that} \quad & H(\mathbf{v}, c) \text{ separates the training data classes} \end{aligned}$$

We can express that as:

$$\begin{aligned} \text{maximize} \quad & d(H(\mathbf{v}, c), \text{training data}) \text{ over all } \mathbf{v}, c \\ \text{such that} \quad & \tilde{y}_i \operatorname{sgn}(\langle \mathbf{v}, \tilde{\mathbf{x}}_i \rangle - c) > 0 \end{aligned}$$

This is an example of a so-called *constrained optimization problem*.

Constrained Optimization

Recall that a basic optimization problem searches for an argmument x^* that makes $f(x^*))$ as small as possible.

Contstrains

Suppose we fix some property of x that is either true or false (e.g. "$x > 0$"). The problem

among all x that satisfy the property, find the one that makes f as small as possible

is called a **constrained optimization problem**. The property is called the **constrained**.

Customary notation

If we call the property A, say, this is often written as:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \text{ satisifies } A \end{aligned}$$

HOW CONSTRAINED PROBLEMS ARE SOLVED

Idea

- An optimization algorithm tries to make f as small as possible.
- We have exclude values of x that violate the constraint.
- Solution: Change the function f so that it is very large at values x that should be excluded.

Implementation

- Choose a function $\beta(x)$ that is very large for all x that violate the constraint, and 0 at those x that are permitted.
- Add β to f: Minimize $f + \beta$ instead of f.
- Remember: We should not introduce jumps, so g should transition smoothly from 0 to "very large".

For example

Say we want to minimize f. For another function g, we impose the constraint $g(x) < 0$.

$$\min f(x) \qquad \text{s.t.} \quad g(x) < 0$$

The constraint $g(x) < 0$ be expressed as an indicator function of $g(x) \geq 0$:

$$\min f(x) + const. \cdot \mathbb{I}_{[0,\infty)}(g(x))$$

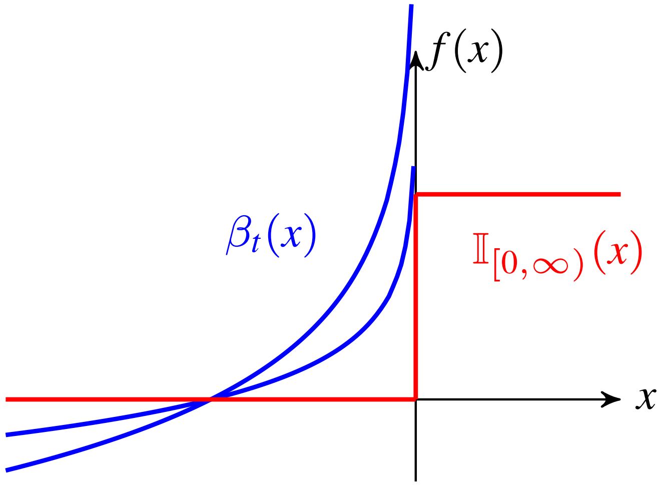
The constant must be chosen large enough to enforce the constraint.

Adding a Smooth Function

Choice of the function we add

- The indicator function jumps, which we know is not useful for optimization. We replace it by a smooth function.
- A common choice is

$$\beta_t(x) := -\frac{1}{t}\log(-x)\ .$$



In the example above

To solve $\min f(x)$ subject to $g(x) < 0$, we apply gradient descent to

$$f(x) + \beta_t(g(x))\ .$$

The value t is a "tuning parameter" of the optimization method.

Constrained Optimization

Remarks

- If the constraint changes (e.g. to "$g(x) > -3$"), it is easier to modify g than to tinker with β.
- The method above is an example of a principle we have seen before: We express what we want to do in terms of an indicator function, then replace it by something smooth, and apply graident descent.
- Data mining, statistics and machine learning are only a few examples of applications of constrained optimization methods. Much of the research on constrained optimization is driven by operations research and financial engineering.

Support Vector Machine

Maximum margin optimization problem

For n training points $(\tilde{\mathbf{x}}_i, \tilde{y}_i)$ with labels $\tilde{y}_i \in \{-1, 1\}$, solve optimization problem:

$$\begin{aligned} \text{maximize} \quad & d\big(H(\mathbf{v}, c), \{\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n\}\big) \\ \text{s.t.} \quad & \tilde{y}_i(\langle \mathbf{v}_{\mathrm{H}}, \tilde{\mathbf{x}}_i \rangle - c) > 0 \quad \text{for } i = 1, \dots, n \end{aligned}$$

- The first line says: Make sure the plane is a far away from every data point as possible.
- The second line says: Only planes that classify the training data correctly are permitted.

Remarks

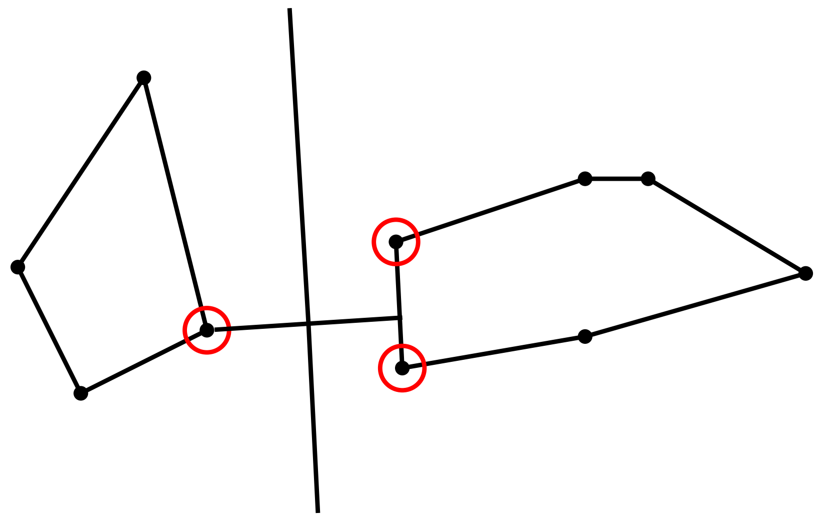
- The classifier obtained by solving this optimization problem is called a **support vector machine**.
- If training data is separable: There is a *unique* solution (in contrast to the perceptron, whose solution is not unique).

Support Vectors

Definition

Those extreme points of the convex hulls which are closest to the hyperplane are called the **support vectors**.

There are at least two support vectors, one in each class.



Implications

- The maximum-margin criterion focuses all attention to the area closest to the decision surface.
- One can show that the computational cost of solving the optimization problem grows quadratically in the number of data points.

SUPPORT VECTOR MACHINES

Advantages

- The SVM often works well for high-dimensional classification.
- It can be generalized to non-linear decision boundaries using a method called the *kernel trick*.
- It can also be generalized to overlapping classes.

Disadvantages

- The quadratic training cost means SVMs cannot be trained on very large data sets.

The support vector machine (with kernel trick) is, aside from a method called a *random forest*, probably the most widely used classifier for non-vision/audio data. For vision and audio data, neural networks dominate applications.