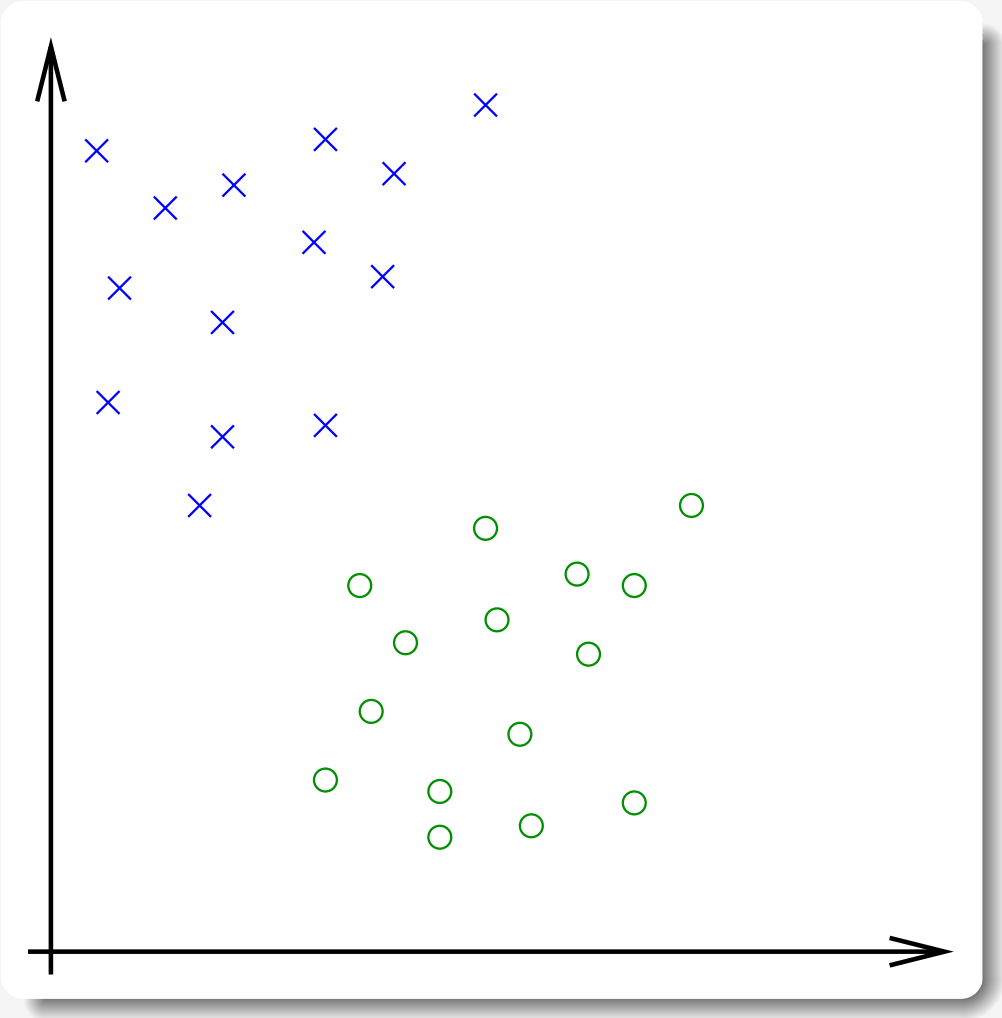
Machine Learning I
Week 8: Support Vector Machines

Martin Felder

Technische Universität München

14 January 2010

Classification Problem Revisited

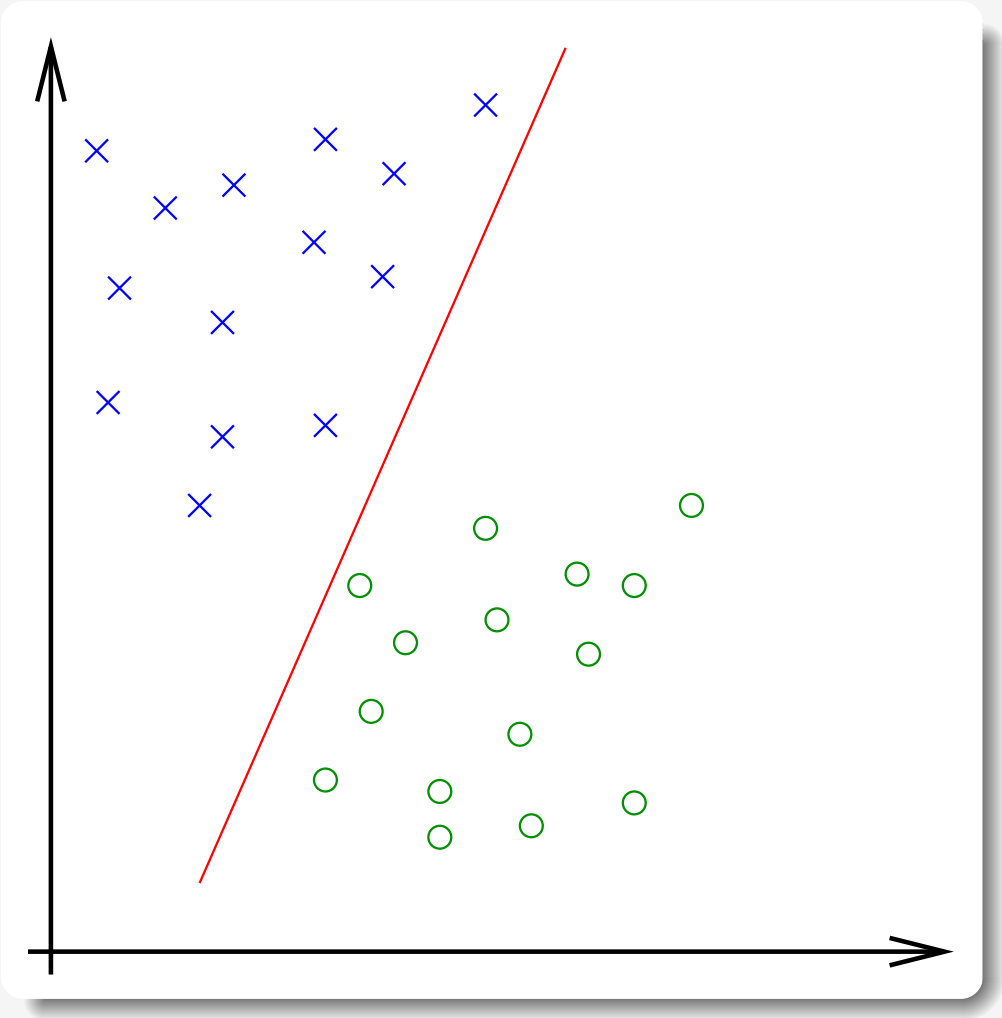


Assume there are two classes of data points in a 2D plane.

Goal: Find the optimum division line between the two classes, such that subsequent points fall onto the correct side.

$\longrightarrow$ Intuitively, a wide margin around the dividing line seems to make this more likely.

Classification Problem Revisited

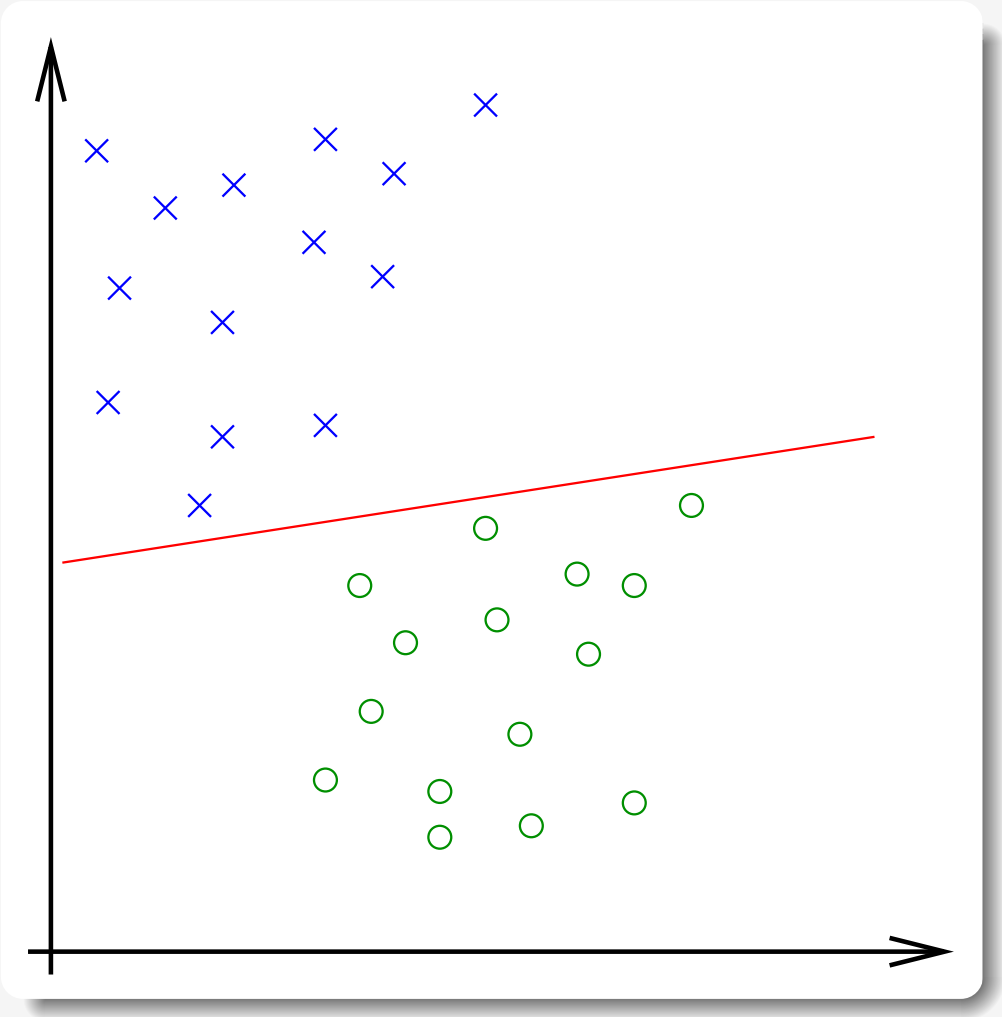


Assume there are two classes of data points in a 2D plane.

Goal: Find the optimum division line between the two classes, such that subsequent points fall onto the correct side.

$\longrightarrow$ Intuitively, a wide margin around the dividing line seems to make this more likely.

Classification Problem Revisited

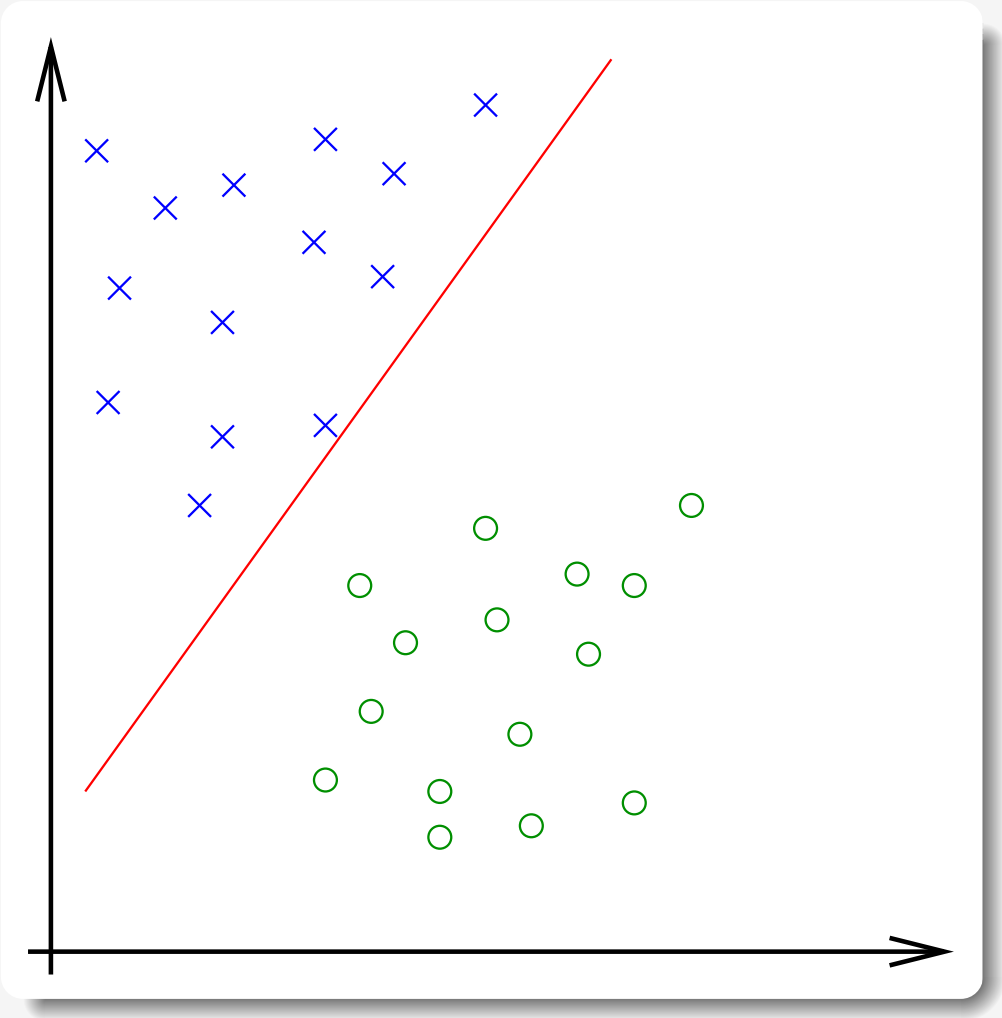


Assume there are two classes of data points in a 2D plane.

Goal: Find the optimum division line between the two classes, such that subsequent points fall onto the correct side.

$\longrightarrow$ Intuitively, a wide margin around the dividing line seems to make this more likely.

Classification Problem Revisited

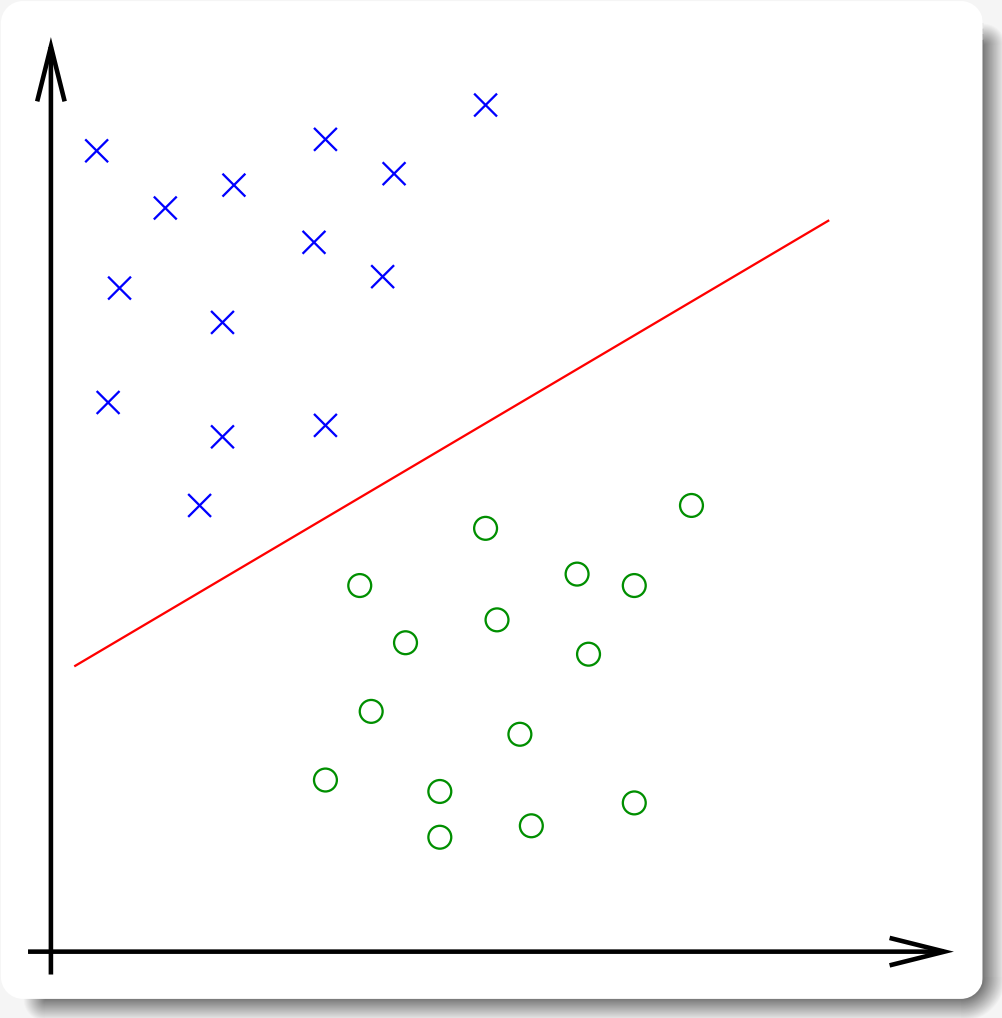


Assume there are two classes of data points in a 2D plane.

Goal: Find the optimum division line between the two classes, such that subsequent points fall onto the correct side.

$\longrightarrow$ Intuitively, a wide margin around the dividing line seems to make this more likely.

Classification Problem Revisited

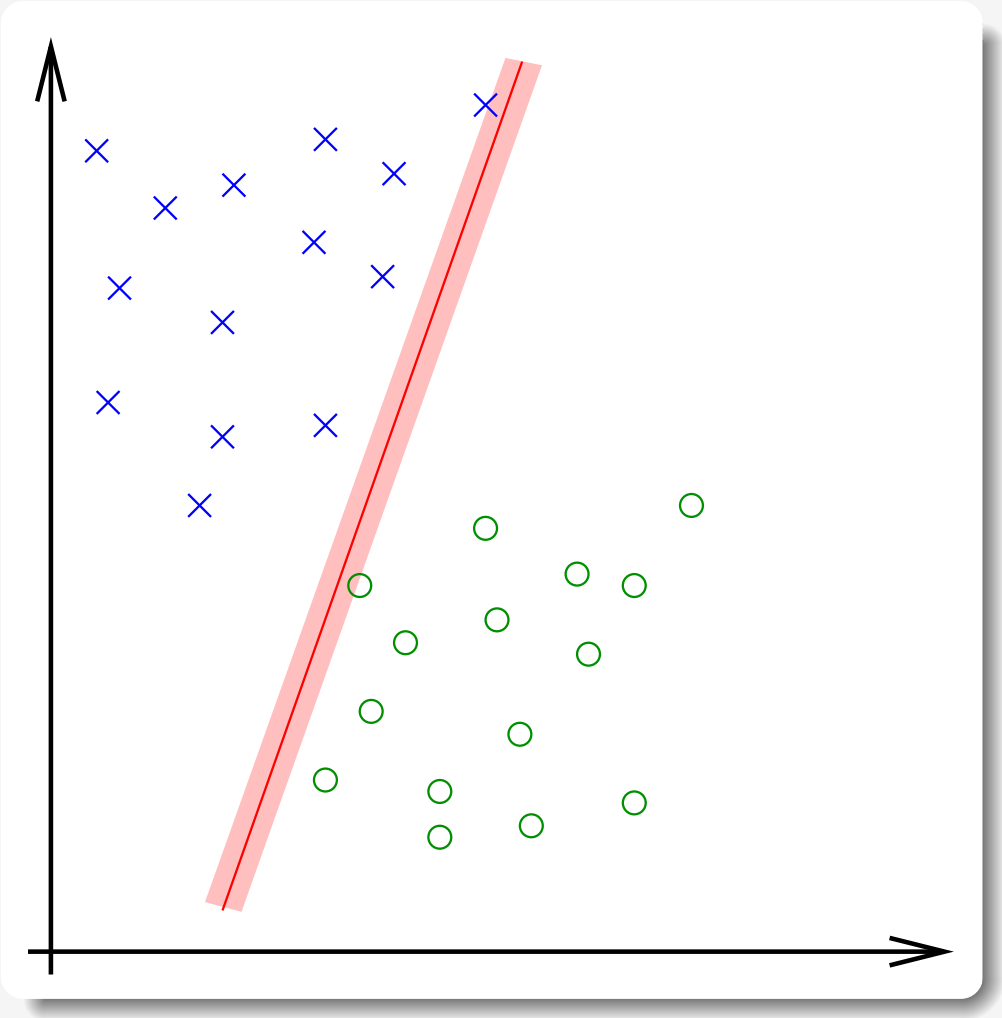


Assume there are two classes of data points in a 2D plane.

Goal: Find the optimum division line between the two classes, such that subsequent points fall onto the correct side.

$\longrightarrow$ Intuitively, a wide margin around the dividing line seems to make this more likely.

Classification Problem Revisited

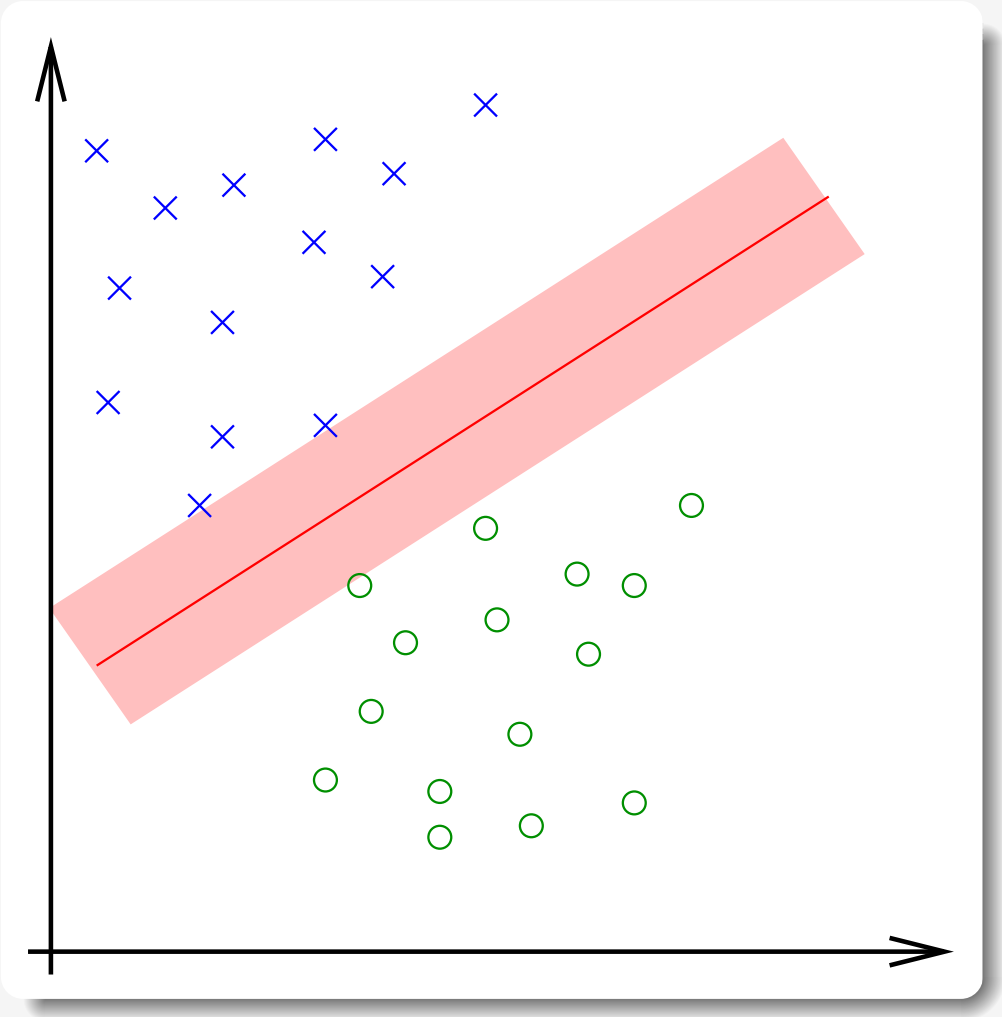


Assume there are two classes of data points in a 2D plane.

Goal: Find the optimum division line between the two classes, such that subsequent points fall onto the correct side.

$\longrightarrow$ Intuitively, a wide margin around the dividing line seems to make this more likely.

Classification Problem Revisited



Assume there are two classes of data points in a 2D plane.

Goal: Find the optimum division line between the two classes, such that subsequent points fall onto the correct side.

$\longrightarrow$ Intuitively, a wide margin around the dividing line seems to make this more likely.

Formal Problem Definition

A hyperplane (i.e. our dividing line, in 2D) can be defined by:

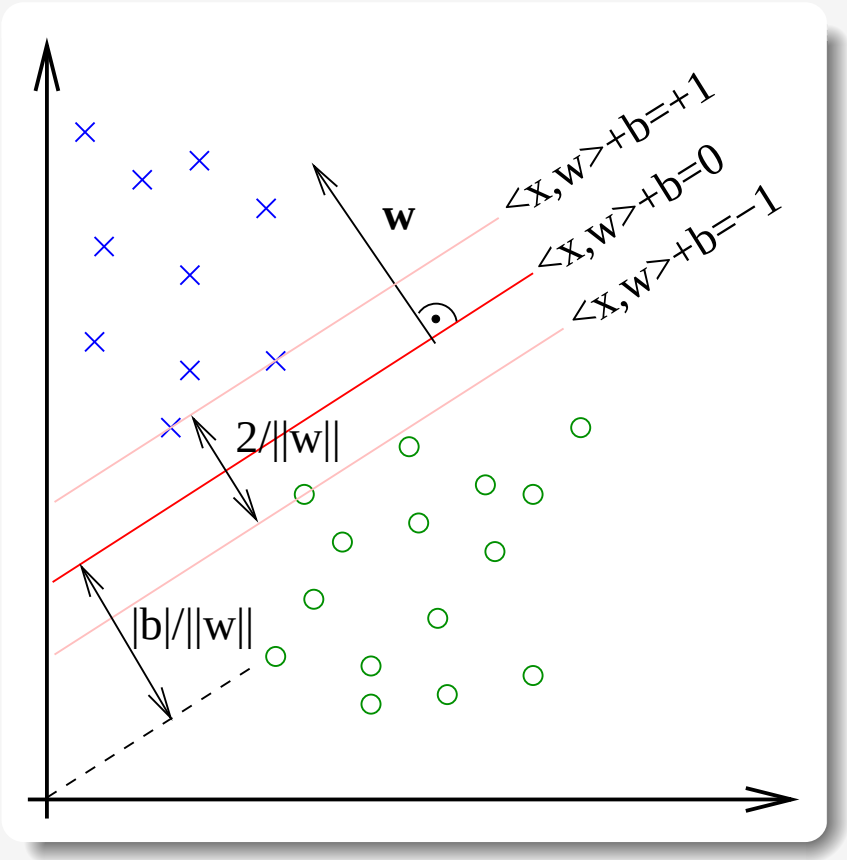
$$\mathbf{x}^T\mathbf{w} + b = 0$$

$\mathbf{w}$ is normal to the hyperplane,
b is the offset,
$\mathbf{x}$ are the points on the hyperplane,

The classifier is then:

$$\text{class}(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T\mathbf{x} + b\right)$$

Note also: The distance from the origin is $|b|/\|\mathbf{w}\|$.



CogBotLab
Machine Learning & Cognitive Robotics

Problem Definition

Let $\mathbf{x}_i$ be the ith of N data points, and $t_i \in \{-1, 1\}$ the class assigned to $\mathbf{x}_i$.

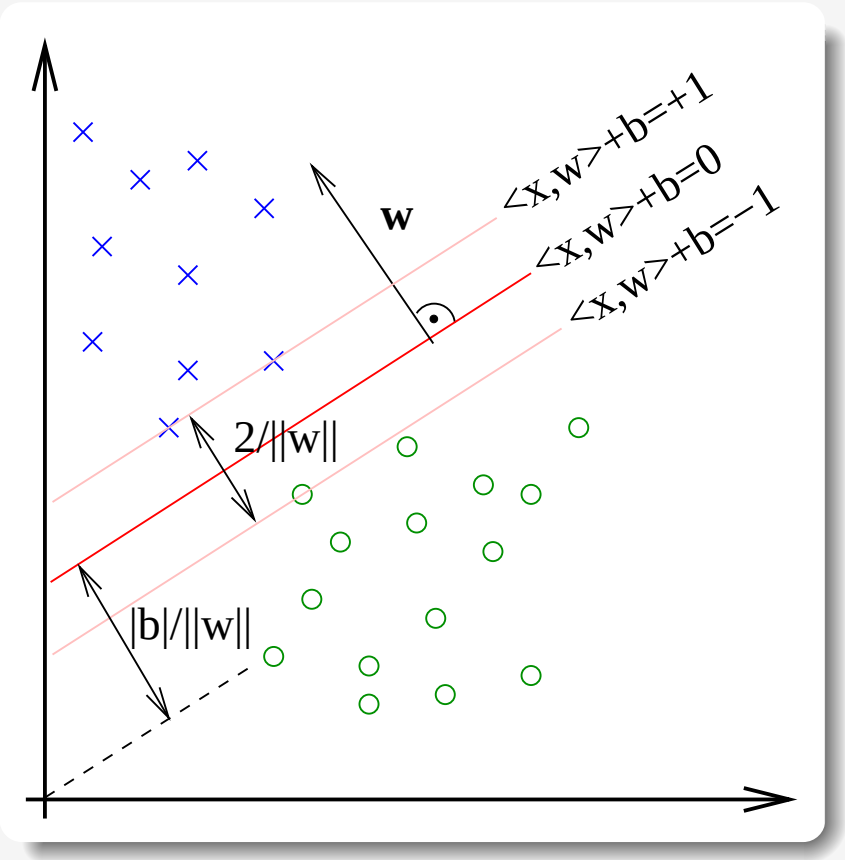
To impose our margin, we state the following constraints $\forall i = 1, \ldots, N$:

$$\mathbf{x}^\mathsf{T}\mathbf{w} + b \geq +1 \quad \text{for } t_i = +1$$

$$\mathbf{x}^\mathsf{T}\mathbf{w} + b \leq -1 \quad \text{for } t_i = -1$$

which can be condensed into

$$t_i(\mathbf{x}_i^\mathsf{T}\mathbf{w} + b) \geq 1$$



CogBotLab
Machine Learning & Cognitive Robotics

Determining the Margin

Now, given the parallel hyperplanes

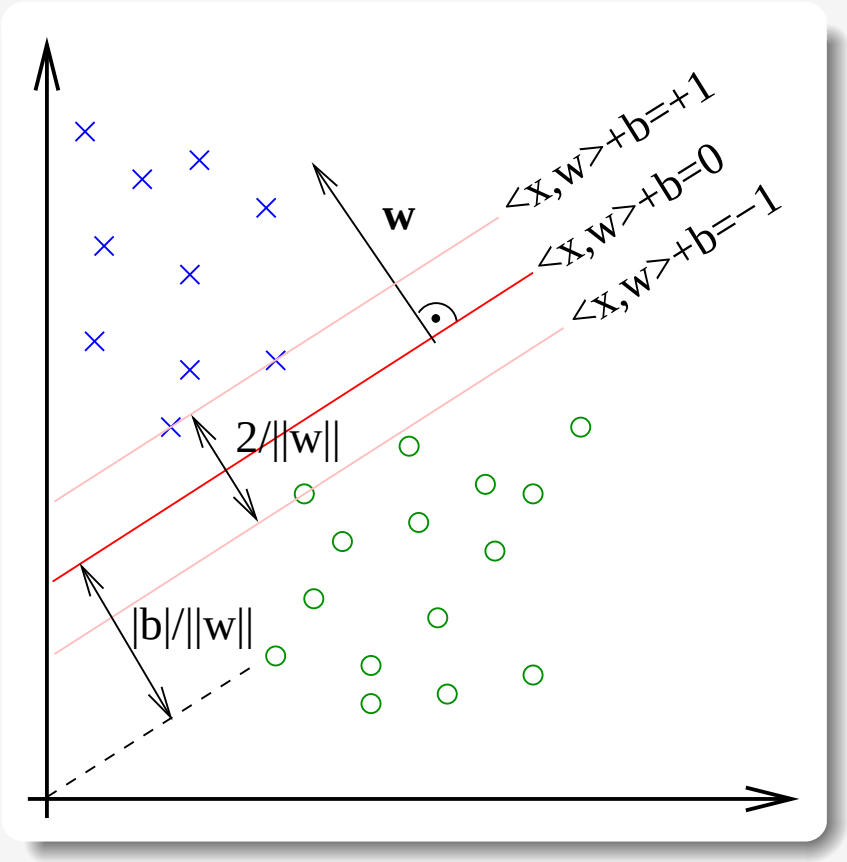
$$\mathbf{x}^T\mathbf{w} + b = \pm 1$$

with the distance to the origin

$$\frac{|\pm 1 - b|}{\|w\|},$$

yields the margin by simple subtraction:

$$\frac{|1-b|}{\|w\|} - \frac{|-1-b|}{\|w\|} = \frac{2}{\|w\|}$$



By definition, no training points occur between the two hyperplanes.

Optimization Problem

Maximizing $2/||\mathbf{w}||$, is equivalent to minimizing $\|\mathbf{w}\|^2/2$. We can formulate this as quadratic programming problem:

> Minimize $\frac{1}{2}\|\mathbf{w}\|^2$, with constraints
>
> $$t_i(\mathbf{x}_i^T\mathbf{w} + b) - 1 \geq 0$$

In principle, QP problems are easy to solve using standard tools, but

- satisfying the constraints in this formulation is a pain, and
- for reasons that will become apparent, we want training data to appear only as dot products in the entire algorithm.

Optimization Problem

Maximizing $2/||\mathbf{w}||$, is equivalent to minimizing $\|\mathbf{w}\|^2/2$. We can formulate this as quadratic programming problem:

Minimize $\frac{1}{2}\|\mathbf{w}\|^2$, with constraints

$$t_i(\mathbf{x}_i^T\mathbf{w} + b) - 1 \geq 0$$

In principle, QP problems are easy to solve using standard tools, but

- satisfying the constraints in this formulation is a pain, and
- for reasons that will become apparent, we want training data to appear only as dot products in the entire algorithm.

Optimization Problem

Maximizing $2/||\mathbf{w}||$, is equivalent to minimizing $\|\mathbf{w}\|^2/2$. We can formulate this as quadratic programming problem:

Minimize $\frac{1}{2}\|\mathbf{w}\|^2$, with constraints

$$t_i(\mathbf{x}_i^T\mathbf{w} + b) - 1 \geq 0$$

In principle, QP problems are easy to solve using standard tools, but

- satisfying the constraints in this formulation is a pain, and
- for reasons that will become apparent, we want training data to appear only as dot products in the entire algorithm.

Lagrange formulation

We reformulate the problem by defining a Lagrangian, using Lagrange Multipliers to provide the constraints.

Rule: *"For constraints of the form $c \geq 0$, subtract λc with $\lambda \geq 0$ from the objective function."*

Ok, let's do that. We get:

Minimize

$$L_P = \frac{1}{2}||\mathbf{w}||^2 - \sum_i \alpha_i t_i(\mathbf{x}_i^T \mathbf{w} + b) + \sum_i \alpha_i$$

with respect to $\mathbf{w}, b$, subject to the constraints

$$\alpha_i \geq 0$$

$$\frac{\partial L_P}{\partial \alpha_i} = t_i(\mathbf{x}_i^T \mathbf{w} + b) + 1 = 0$$

Lagrange formulation

We reformulate the problem by defining a Lagrangian, using Lagrange Multipliers to provide the constraints.

Rule: *"For constraints of the form $c \geq 0$, subtract λc with $\lambda \geq 0$ from the objective function."*

Ok, let's do that. We get:

Minimize

$$L_P = \frac{1}{2}||\mathbf{w}||^2 - \sum_i \alpha_i t_i(\mathbf{x}_i^T \mathbf{w} + b) + \sum_i \alpha_i$$

with respect to $\mathbf{w}, b$, subject to the constraints

$$\alpha_i \geq 0$$

$$\frac{\partial L_P}{\partial \alpha_i} = t_i(\mathbf{x}_i^T \mathbf{w} + b) + 1 = 0$$

Lagrange formulation

- Notice how we have shifted the inequality conditions to the αs.
- This is a convex QP problem, since the objective function is convex, and linear constraints always define a convex set of points.
- Optimization theory has shown that under these conditions, we can equivalently solve the Wolfe dual problem:

Minimize

$$L_P = \frac{1}{2}||\mathbf{w}||^2 - \sum_i \alpha_i t_i(\mathbf{x}_i^T \mathbf{w} + b) + \sum_i \alpha_i$$

with respect to $\boldsymbol{\alpha}$, subject to the constraints

$$\alpha_i \geq 0$$

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i t_i \mathbf{x}_i = 0 \quad (1)$$

$$\frac{\partial L_P}{\partial b} = -\sum_i \alpha_i t_i = 0 \quad (2)$$

Lagrange formulation – Dual

Since this is a convex problem, there is a single unique maximum. Substituting the dual problem constraints (1) and (2) back into L_P allows us to reformulate:

Maximize

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\mathsf{T} \mathbf{x}_j$$

with respect to $\boldsymbol{\alpha}$, subject to the constraints

$$\alpha_i \geq 0$$

$$\mathbf{w} - \sum_i \alpha_i t_i \mathbf{x}_i = 0$$

$$-\sum_i \alpha_i t_i = 0$$

Lagrange formulation – Dual

Notes:

- Solving the dual problem often much simpler than primal problem.
- $H_{ij} := t_i t_j \mathbf{x}_i^T \mathbf{x}_j$ defines the Hessian, which is usually positive semidefinite.
- b is not explicitly given by the solution condition. After the α_i are found, it can be calculated as

$$b = \frac{1}{t_i} - \mathbf{w}^\mathsf{T} \mathbf{x}_i,$$

where an average over all α_i is usually taken for numerical stability reasons.

Problem: Wait a minute, what's so great about this? Inverting the Hessian scales as $\mathcal{O}(N^3)$ worst case!

$\longrightarrow$ let's look at the optimality result in more detail

Lagrange formulation – Dual

Notes:

- Solving the dual problem often much simpler than primal problem.
- $H_{ij} := t_i t_j \mathbf{x}_i^T \mathbf{x}_j$ defines the Hessian, which is usually positive semidefinite.
- b is not explicitly given by the solution condition. After the α_i are found, it can be calculated as

$$b = \frac{1}{t_i} - \mathbf{w}^\mathsf{T} \mathbf{x}_i,$$

where an average over all α_i is usually taken for numerical stability reasons.

Problem: Wait a minute, what's so great about this? Inverting the Hessian scales as $\mathcal{O}(N^3)$ worst case!

$\longrightarrow$ let's look at the optimality result in more detail

Karush-Kuhn-Tucker Conditions

It can be proven that optimality is given for our convex QP iff the Karush-Kuhn-Tucker conditions hold:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i t_i \mathbf{x}_i = 0$$

$$\frac{\partial L_P}{\partial b} = -\sum_i \alpha_i t_i = 0$$

$$t_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) - 1 \geq 0 \quad \forall i$$

$$\alpha_i \geq 0$$

$$\alpha_i(t_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) - 1) = 0 \quad \forall i$$

Karush-Kuhn-Tucker Conditions

It can be proven that optimality is given for our convex QP iff the Karush-Kuhn-Tucker conditions hold:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i t_i \mathbf{x}_i = 0$$

$$\frac{\partial L_P}{\partial b} = -\sum_i \alpha_i t_i = 0$$

$$t_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) - 1 \geq 0 \quad \forall i$$

$$\alpha_i \geq 0$$

$$\alpha_i(t_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) - 1) = 0 \quad \forall i$$

Support Vectors

$$\alpha_i(t_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) = 0 \quad \forall i$$

- The Lagrange-Multiplicators α_i are
  $\alpha_i = 0$ if $\mathbf{x}_i$ doesn't contribute to the solution, and
  $\alpha_i > 0$ if $\mathbf{x}_i$ does influence the hyperplane.
- $\mathbf{x}_i$ with $\alpha_i > 0$ are called the support vectors.
- Note that in the dual formulation we had:
  $\mathbf{w} = \sum_i \alpha_i t_i \mathbf{x}_i$
- Hence to calculate $\mathbf{w}$, it suffices to know all $\alpha_i > 0$, with corresponding $\mathbf{x}_i$ and class t_i.
- This means most data points are irrelevant to the solution.

Support Vectors

$$\alpha_i(t_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) = 0 \quad \forall i$$

- The Lagrange-Multiplicators α_i are
  $\alpha_i = 0$ if $\mathbf{x}_i$ doesn't contribute to the solution, and
  $\alpha_i > 0$ if $\mathbf{x}_i$ does influence the hyperplane.
- $\mathbf{x}_i$ with $\alpha_i > 0$ are called the support vectors.
- Note that in the dual formulation we had:
  $\mathbf{w} = \sum_i \alpha_i t_i \mathbf{x}_i$
- Hence to calculate $\mathbf{w}$, it suffices to know all $\alpha_i > 0$, with corresponding $\mathbf{x}_i$ and class t_i.
- This means most data points are irrelevant to the solution.

Support Vectors

$$\alpha_i(t_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) = 0 \quad \forall i$$

- The Lagrange-Multiplicators α_i are
  $\alpha_i = 0$ if $\mathbf{x}_i$ doesn't contribute to the solution, and
  $\alpha_i > 0$ if $\mathbf{x}_i$ does influence the hyperplane.
- $\mathbf{x}_i$ with $\alpha_i > 0$ are called the support vectors.
- Note that in the dual formulation we had:
  $\mathbf{w} = \sum_i \alpha_i t_i \mathbf{x}_i$
- Hence to calculate $\mathbf{w}$, it suffices to know all $\alpha_i > 0$, with corresponding $\mathbf{x}_i$ and class t_i.
- This means most data points are irrelevant to the solution.

Support Vectors

$$\alpha_i(t_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) = 0 \quad \forall i$$

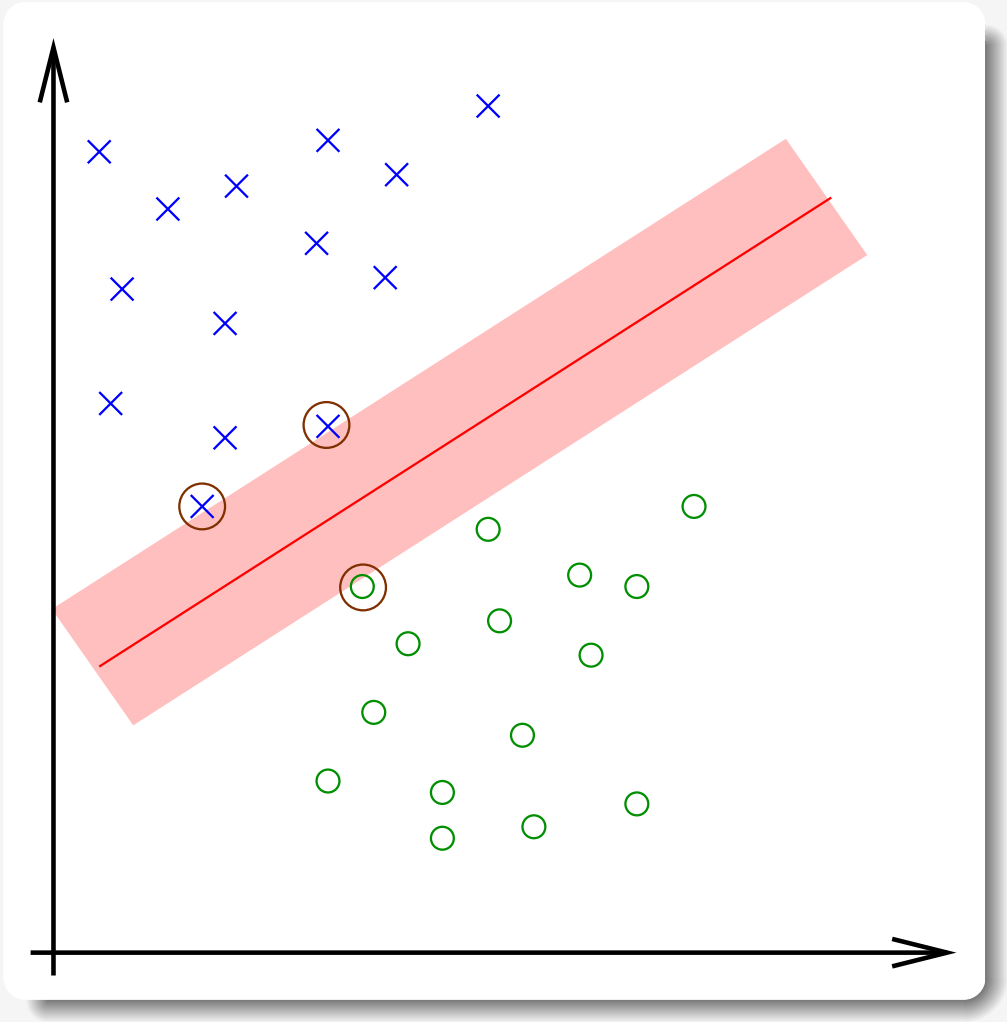
- The Lagrange-Multiplicators α_i are
  $\alpha_i = 0$ if $\mathbf{x}_i$ doesn't contribute to the solution, and
  $\alpha_i > 0$ if $\mathbf{x}_i$ does influence the hyperplane.
- $\mathbf{x}_i$ with $\alpha_i > 0$ are called the support vectors.
- Note that in the dual formulation we had:
  $\mathbf{w} = \sum_i \alpha_i t_i \mathbf{x}_i$
- Hence to calculate $\mathbf{w}$, it suffices to know all $\alpha_i > 0$, with corresponding $\mathbf{x}_i$ and class t_i.
- This means most data points are irrelevant to the solution.

Support Vectors

$$\alpha_i(t_i(\mathbf{w}^T\mathbf{x}_i + b) - 1) = 0 \quad \forall i$$

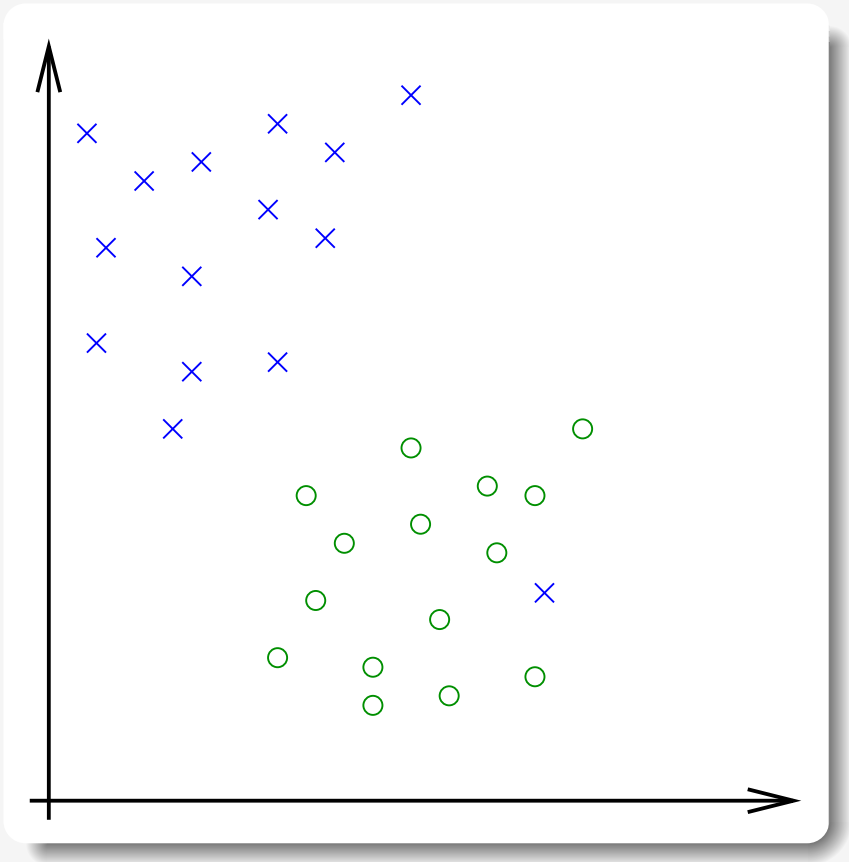
- The Lagrange-Multiplicators α_i are
  $\alpha_i = 0$ if $\mathbf{x}_i$ doesn't contribute to the solution, and
  $\alpha_i > 0$ if $\mathbf{x}_i$ does influence the hyperplane.
- $\mathbf{x}_i$ with $\alpha_i > 0$ are called the support vectors.
- Note that in the dual formulation we had:
  $\mathbf{w} = \sum_i \alpha_i t_i \mathbf{x}_i$
- Hence to calculate $\mathbf{w}$, it suffices to know all $\alpha_i > 0$, with corresponding $\mathbf{x}_i$ and class t_i.
- This means most data points are irrelevant to the solution.

Support Vectors



Separability

What if data is not linearly separable?



$\Longrightarrow$ training will not converge!

Separability

- Add a slack variable ξ to the constraints:

$$\mathbf{x}_i^T \mathbf{w} + b \geq +1 - \xi_i \quad \text{for } t_i = +1$$

$$\mathbf{x}_i^T \mathbf{w} + b \leq -1 + \xi_i \quad \text{for } t_i = -1$$

  which can again be joined:

$$t_i(\mathbf{x}_i^T \mathbf{w} + b) - 1 + \xi_i \geq 0$$

$$\xi_i \geq 0$$

- So the objective function to be minimized becomes:

$$\frac{\|\mathbf{w}\|^2}{2} + C \sum_i \xi_i$$

- C must be chosen by the user: Higher C means stronger punishment of misclassification.

Separability

- Add a slack variable ξ to the constraints:

$$\mathbf{x}_i^T \mathbf{w} + b \geq +1 - \xi_i \quad \text{for } t_i = +1$$

$$\mathbf{x}_i^T \mathbf{w} + b \leq -1 + \xi_i \quad \text{for } t_i = -1$$

  which can again be joined:

$$t_i(\mathbf{x}_i^T \mathbf{w} + b) - 1 + \xi_i \geq 0$$

$$\xi_i \geq 0$$

- So the objective function to be minimized becomes:

$$\frac{\|\mathbf{w}\|^2}{2} + C \sum_i \xi_i$$

- C must be chosen by the user: Higher C means stronger punishment of misclassification.

Separability

- Add a slack variable ξ to the constraints:

$$\mathbf{x}_i^T \mathbf{w} + b \geq +1 - \xi_i \quad \text{for } t_i = +1$$

$$\mathbf{x}_i^T \mathbf{w} + b \leq -1 + \xi_i \quad \text{for } t_i = -1$$

  which can again be joined:

$$t_i(\mathbf{x}_i^T \mathbf{w} + b) - 1 + \xi_i \geq 0$$

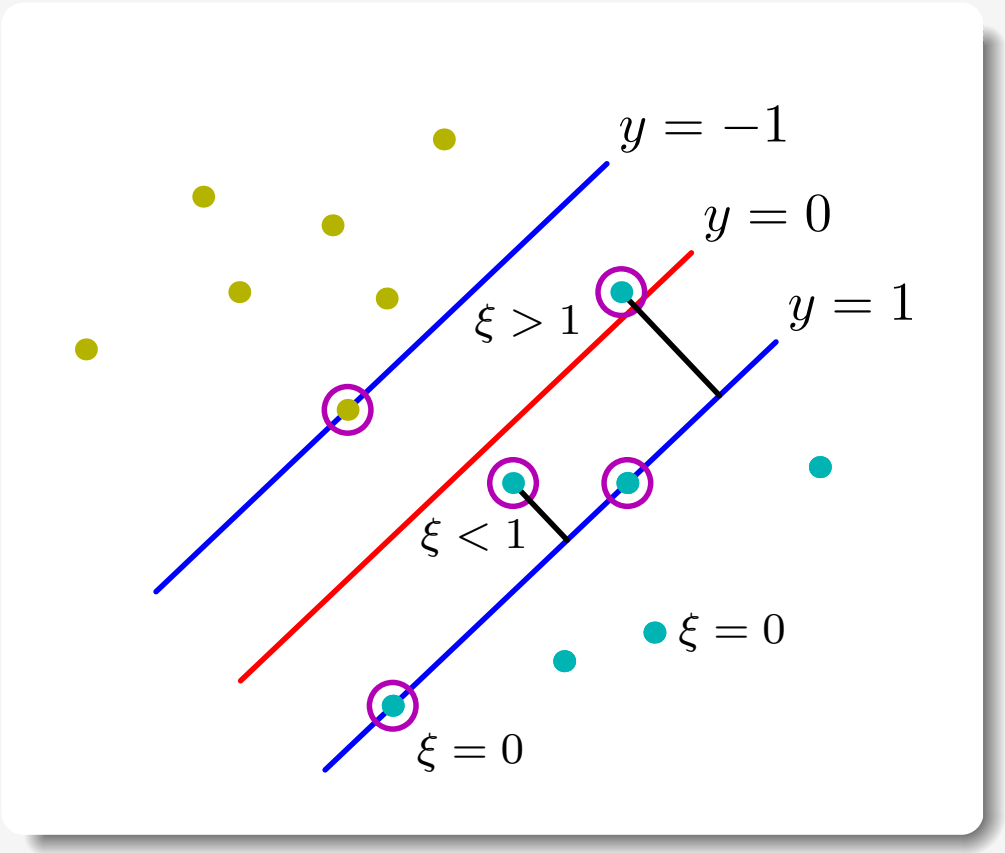
$$\xi_i \geq 0$$

- So the objective function to be minimized becomes:

$$\frac{\|\mathbf{w}\|^2}{2} + C \sum_i \xi_i$$

- C must be chosen by the user: Higher C means stronger punishment of misclassification.

Separability



We add the new constraints with another set of Lagrange multipliers μ_i:

The primal Lagrangian thus becomes:

$$L_P = \frac{\|\mathbf{w}\|^2}{2} + C\sum_i \xi_i - \sum_i \alpha_i(t_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) - 1 + \xi_i) - \sum_i \mu_i\xi_i$$

With the following KKT conditions:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i t_i \mathbf{x}_i = 0 \tag{3}$$

$$\frac{\partial L_P}{\partial b} = -\sum_i \alpha_i t_i = 0 \tag{4}$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \tag{5}$$

$$t_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) - 1 + \xi_i \geq 0 \tag{6}$$

$$\alpha_i \geq 0 \quad \xi_i \geq 0 \quad \mu_i \geq 0 \tag{7}$$

$$\alpha_i(t_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) - 1 + \xi_i) = 0 \tag{8}$$

$$\mu_i\xi_i = 0 \tag{9}$$

Substitution of the KKT equations into the primal yields the dual Lagrangian:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\mathsf{T} \mathbf{x}_j,$$

where still

$$\sum_i \alpha_i t_i = 0.$$

From the conditions $\mu_i \xi_i = 0$ and $C - \alpha_i - \mu_i = 0$ we can also conclude that

$$\xi_i = 0 \quad \text{if} \quad \alpha_i < C.$$

Thus looking at the points where $0 < \alpha_i < C$, Eqn. (8) reverts to the unconstrained case, and the solutions for $\mathbf{w}$ and b still compute as

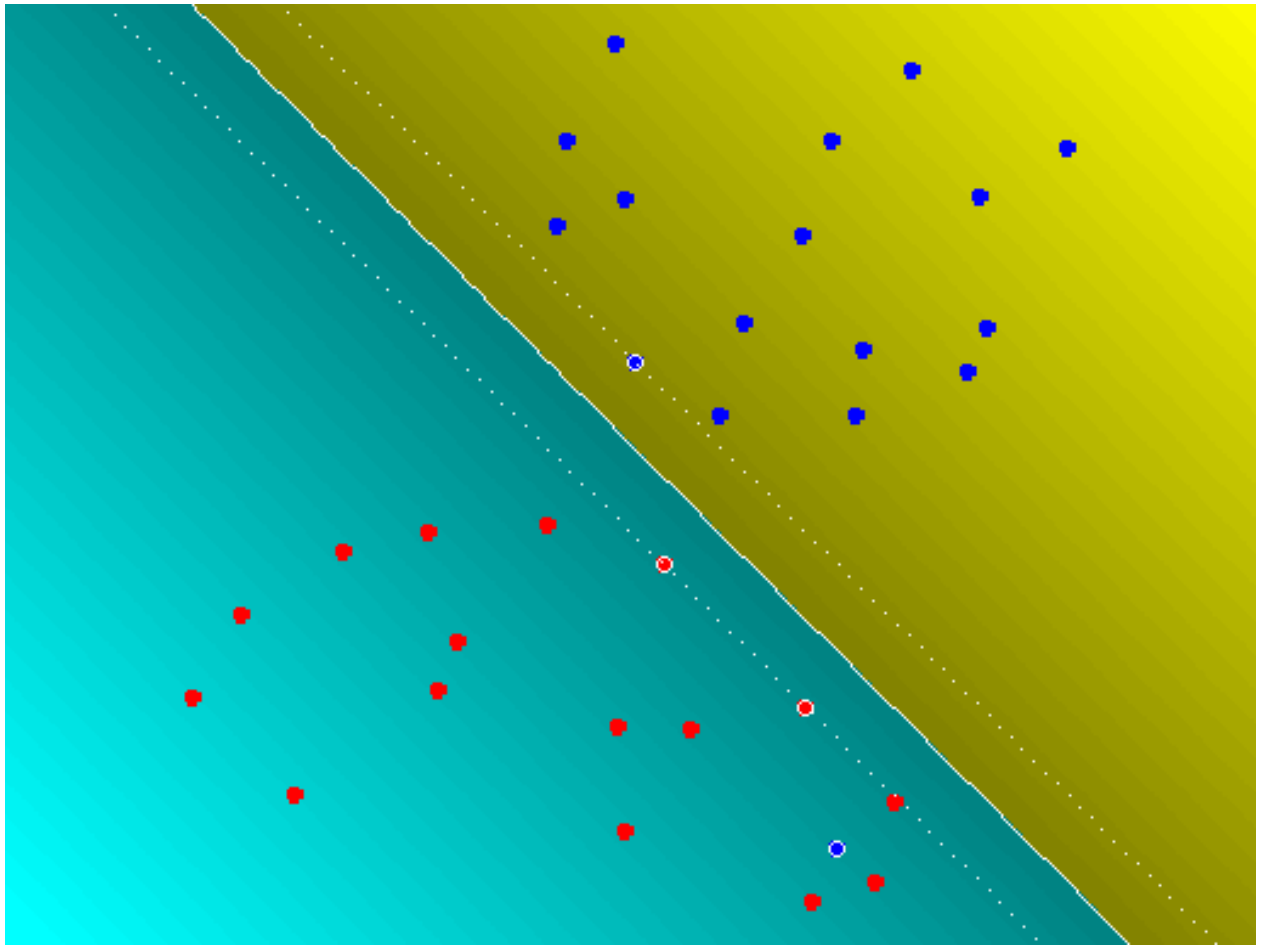
$$\mathbf{w} = \sum_i \alpha_i t_i \mathbf{x}_i$$

$$b = \frac{1}{t_i} - \mathbf{x}_i^\mathsf{T} \mathbf{w}$$

.

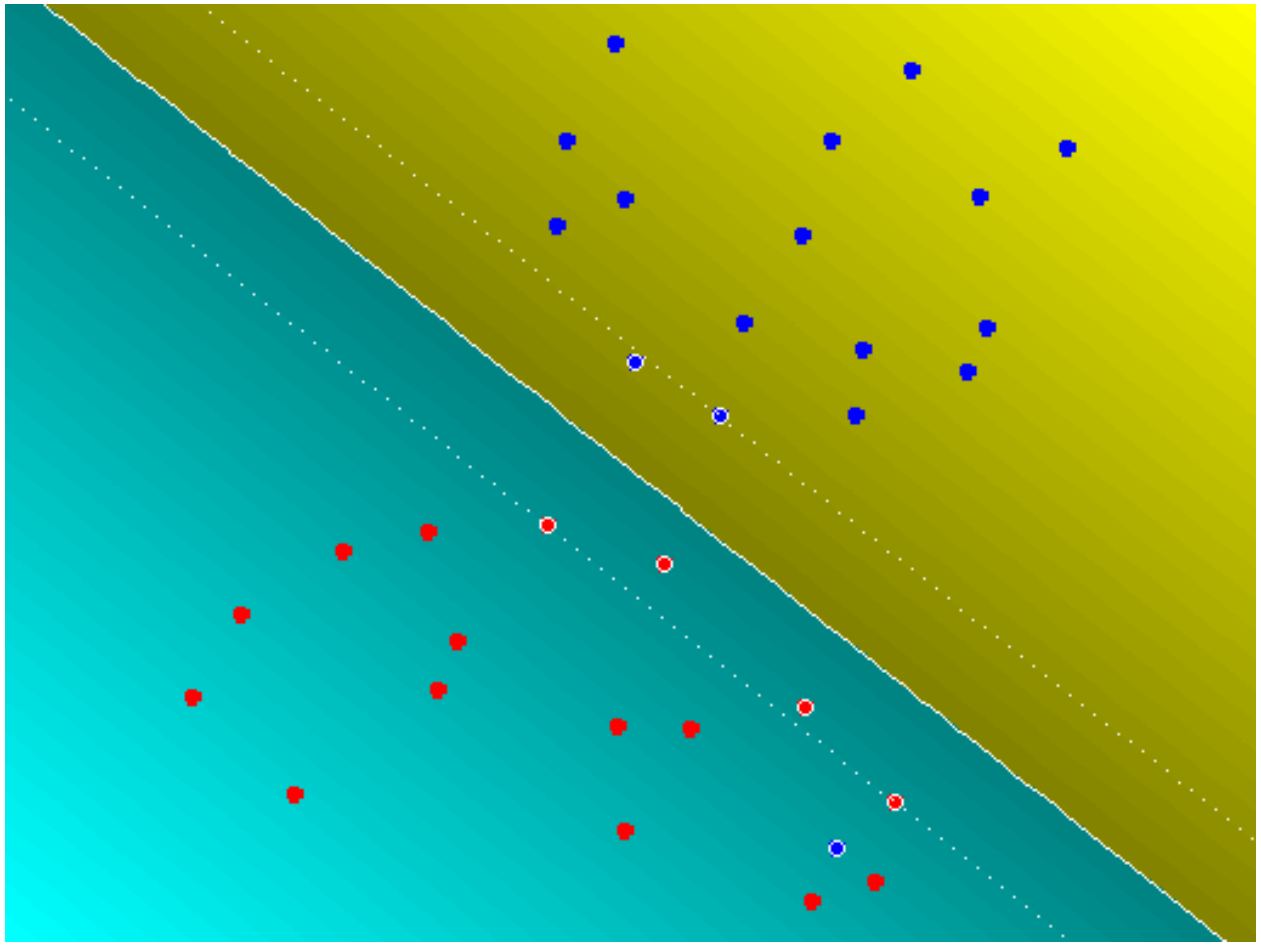
Influence of C

$$C = 100$$



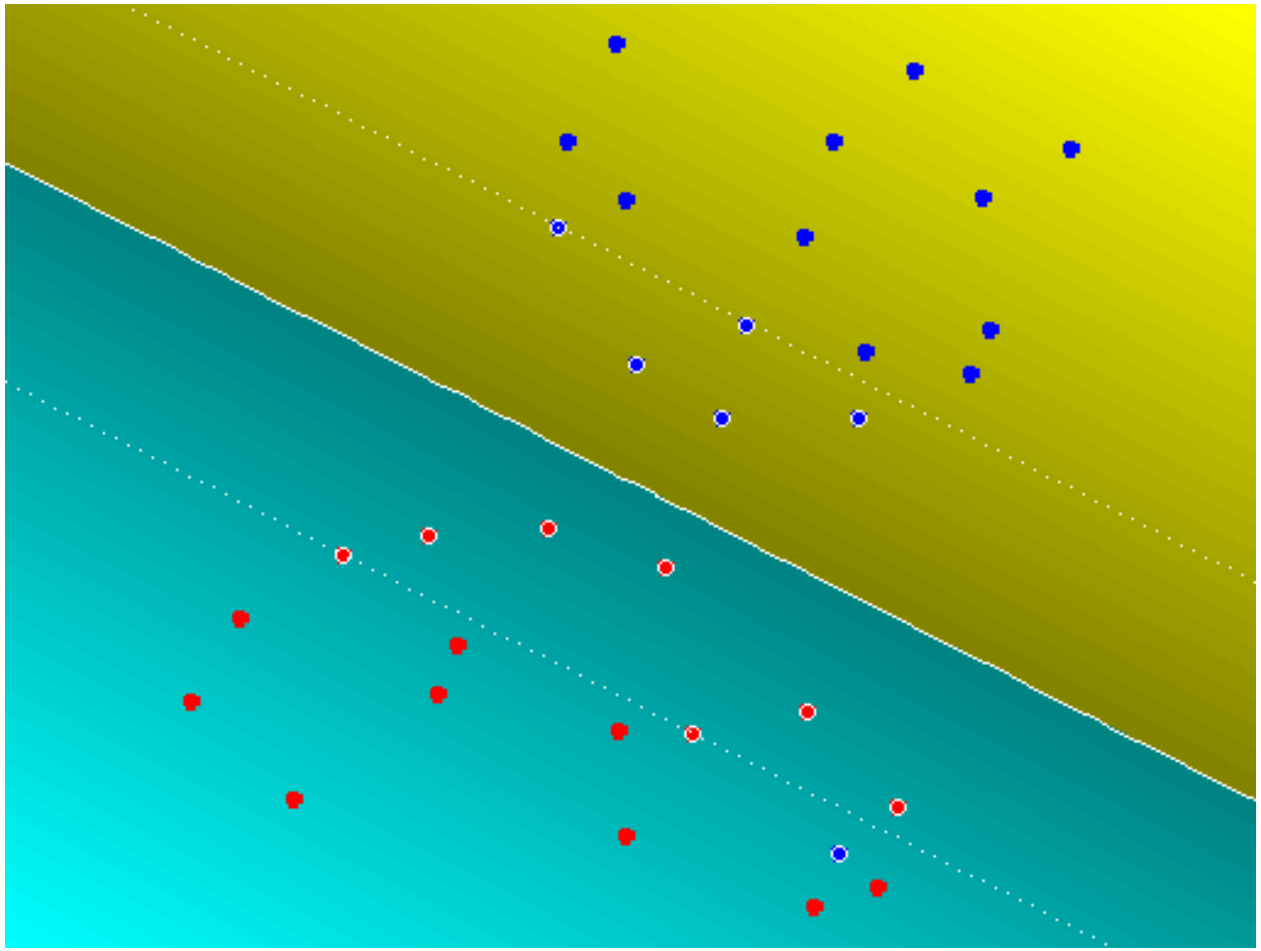
Influence of C

$$C = 10$$



Influence of C

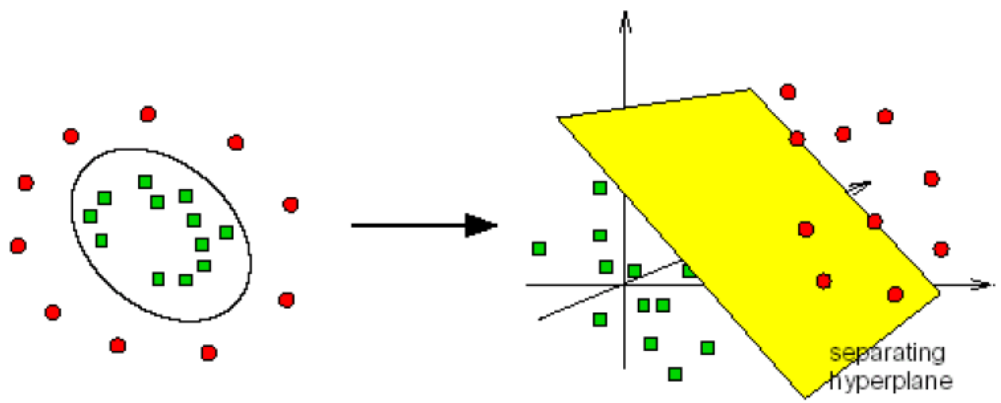
$$C = 1$$



How to build a more general classifier?

- Build model that can produce non-linear boundaries = The Neural Network way
- Stick with the hyperplane, but instead raise the data to a higher dimension ("feature space") by some transformation $\mathbf{\Phi}(\cdot)$ ("feature map") = The SVM way

Boundaries often become simpler in higher dimensions



How to build a more general classifier?

Example: Can't tell from a 2D image which of these stars belong to our galaxy.

However, given distance as a 3rd dimension, you can.

Kernel Trick

- Previously, we had:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\mathsf{T} \mathbf{x}_j,$$

  where the training data $\mathbf{x}$ only appear in a scalar product.

- Hence, only map the scalar product:

$$k(\mathbf{a}, \mathbf{b}) = \phi(\mathbf{a})^\mathsf{T} \phi(\mathbf{b})$$

$$k : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$$

- k is then called the kernel of the transformation.
- This can be orders of magnitude faster that mapping the data directly using $\Phi(\cdot)$!

Definition: Support Vector Machine [?]

An SVM is a maximum separability hyperplane in feature space constructed by means of a kernel function in data space.

Kernel Trick

- Previously, we had:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\mathsf{T} \mathbf{x}_j,$$

  where the training data $\mathbf{x}$ only appear in a scalar product.
- Hence, only map the scalar product:

$$k(\mathbf{a}, \mathbf{b}) = \phi(\mathbf{a})^\mathsf{T} \phi(\mathbf{b})$$

$$k : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$$

- k is then called the kernel of the transformation.
- This can be orders of magnitude faster that mapping the data directly using $\Phi(\cdot)$!

Definition: Support Vector Machine [?]

An SVM is a maximum separability hyperplane in feature space constructed by means of a kernel function in data space.

Kernel Trick

- Previously, we had:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\mathsf{T} \mathbf{x}_j,$$

  where the training data $\mathbf{x}$ only appear in a scalar product.
- Hence, only map the scalar product:

$$k(\mathbf{a}, \mathbf{b}) = \phi(\mathbf{a})^\mathsf{T} \phi(\mathbf{b})$$

$$k : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$$

- k is then called the kernel of the transformation.
- This can be orders of magnitude faster that mapping the data directly using $\Phi(\cdot)$!

Definition: Support Vector Machine [?]

An SVM is a maximum separability hyperplane in feature space constructed by means of a kernel function in data space.

Kernel Trick

- Previously, we had:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \mathbf{x}_i^\mathsf{T} \mathbf{x}_j,$$

  where the training data $\mathbf{x}$ only appear in a scalar product.
- Hence, only map the scalar product:

$$k(\mathbf{a}, \mathbf{b}) = \phi(\mathbf{a})^\mathsf{T} \phi(\mathbf{b})$$

$$k : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$$

- k is then called the kernel of the transformation.
- This can be orders of magnitude faster that mapping the data directly using $\Phi(\cdot)$!

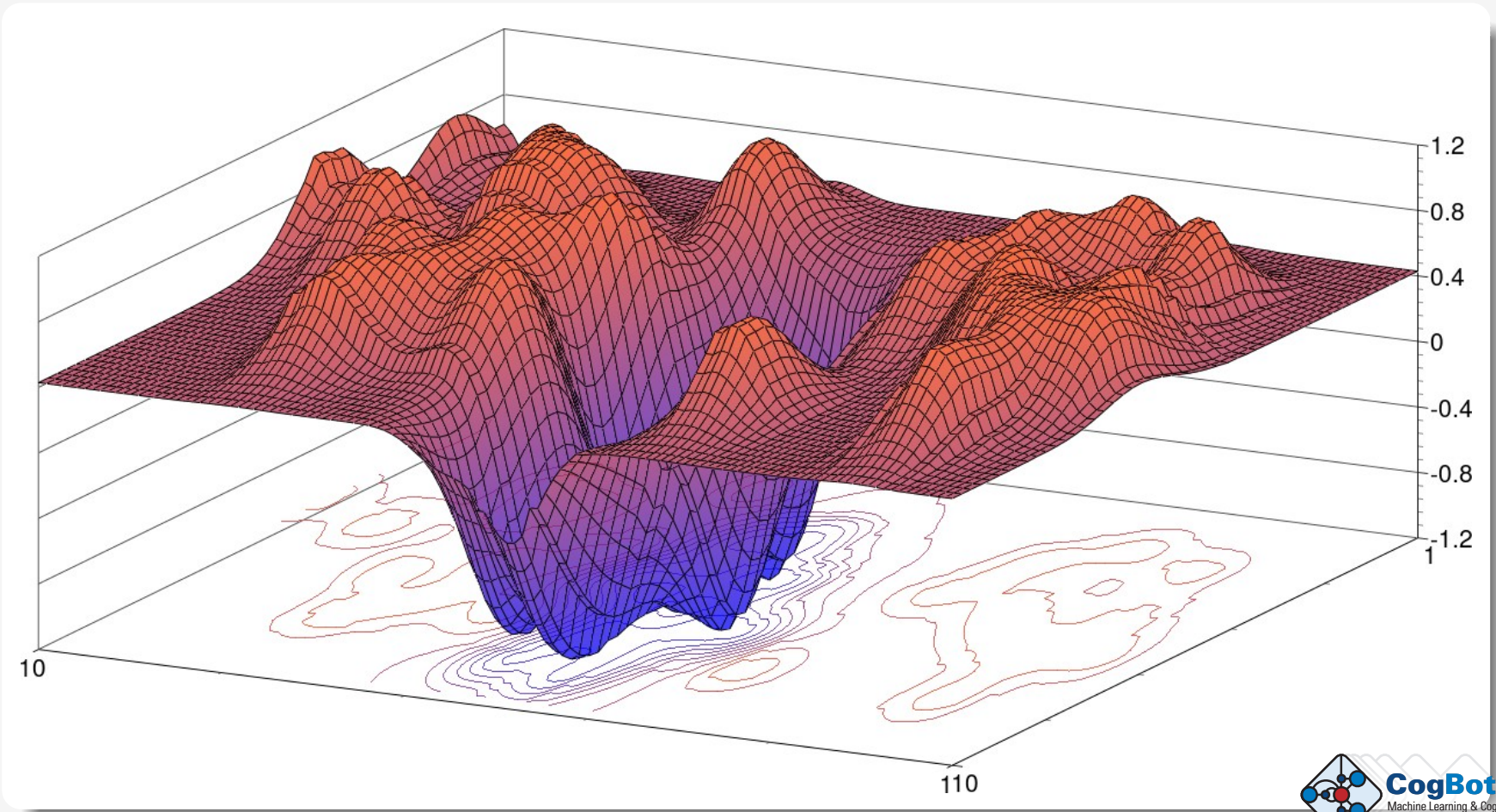
Definition: Support Vector Machine [?]

An SVM is a maximum separability hyperplane in feature space constructed by means of a kernel function in data space.

Gaussian kernel

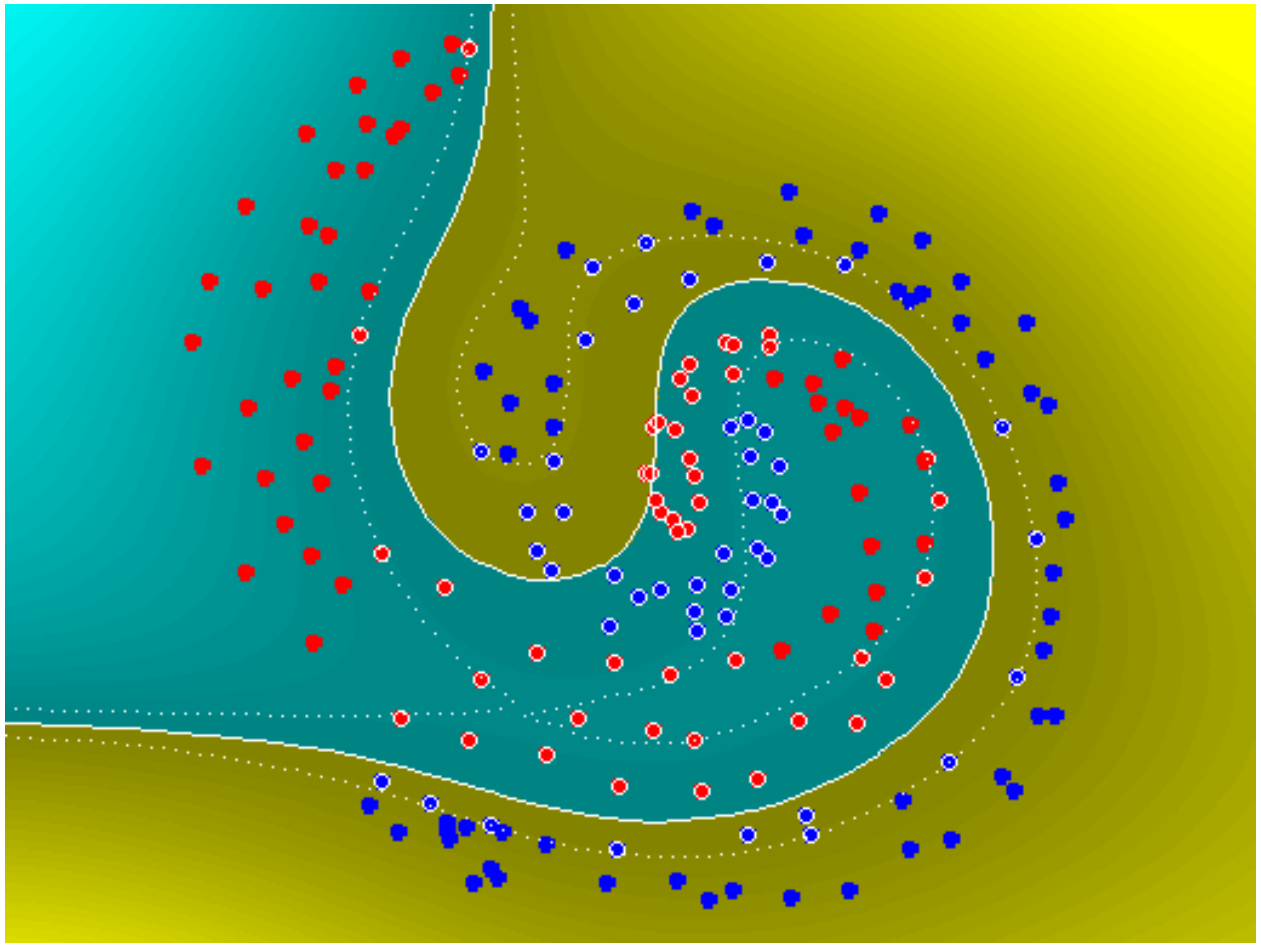
Visualization of Gaussian kernel feature space (distance from decision boundary)

$$\sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$



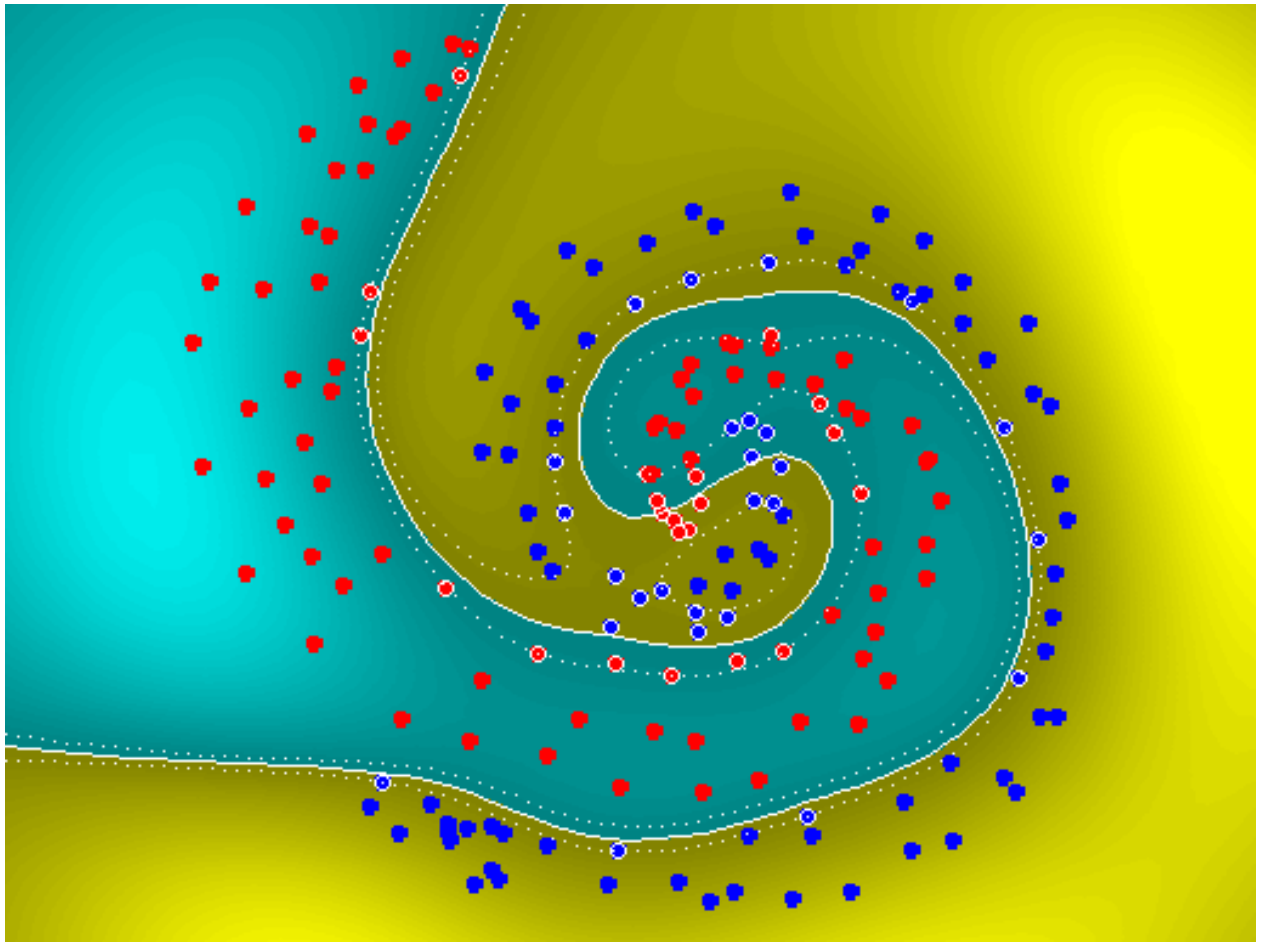
Gaussian kernel

$$\sigma = 1 \quad C = 1000$$



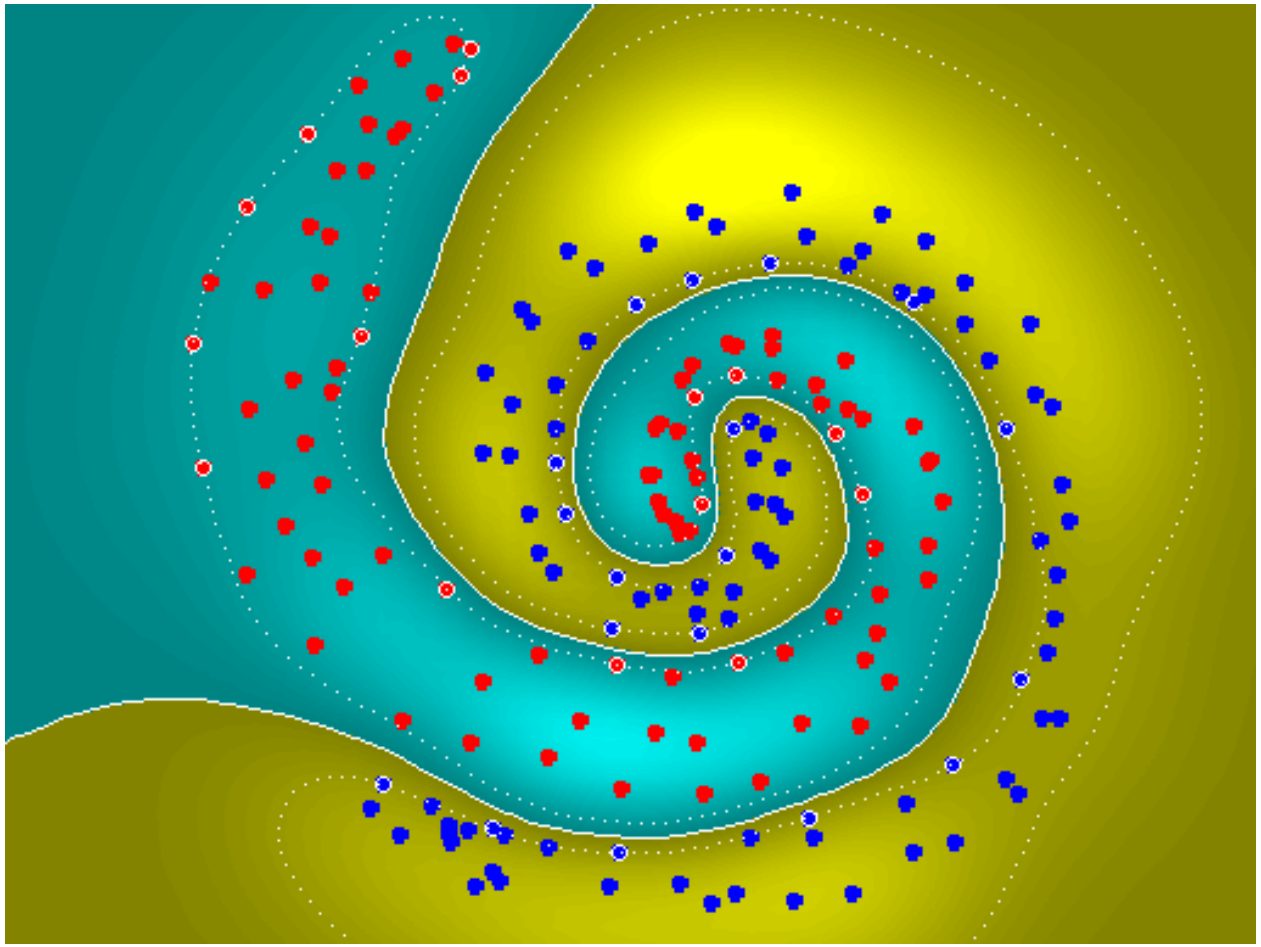
Gaussian kernel

$$\sigma = .5 \quad C = 1000$$



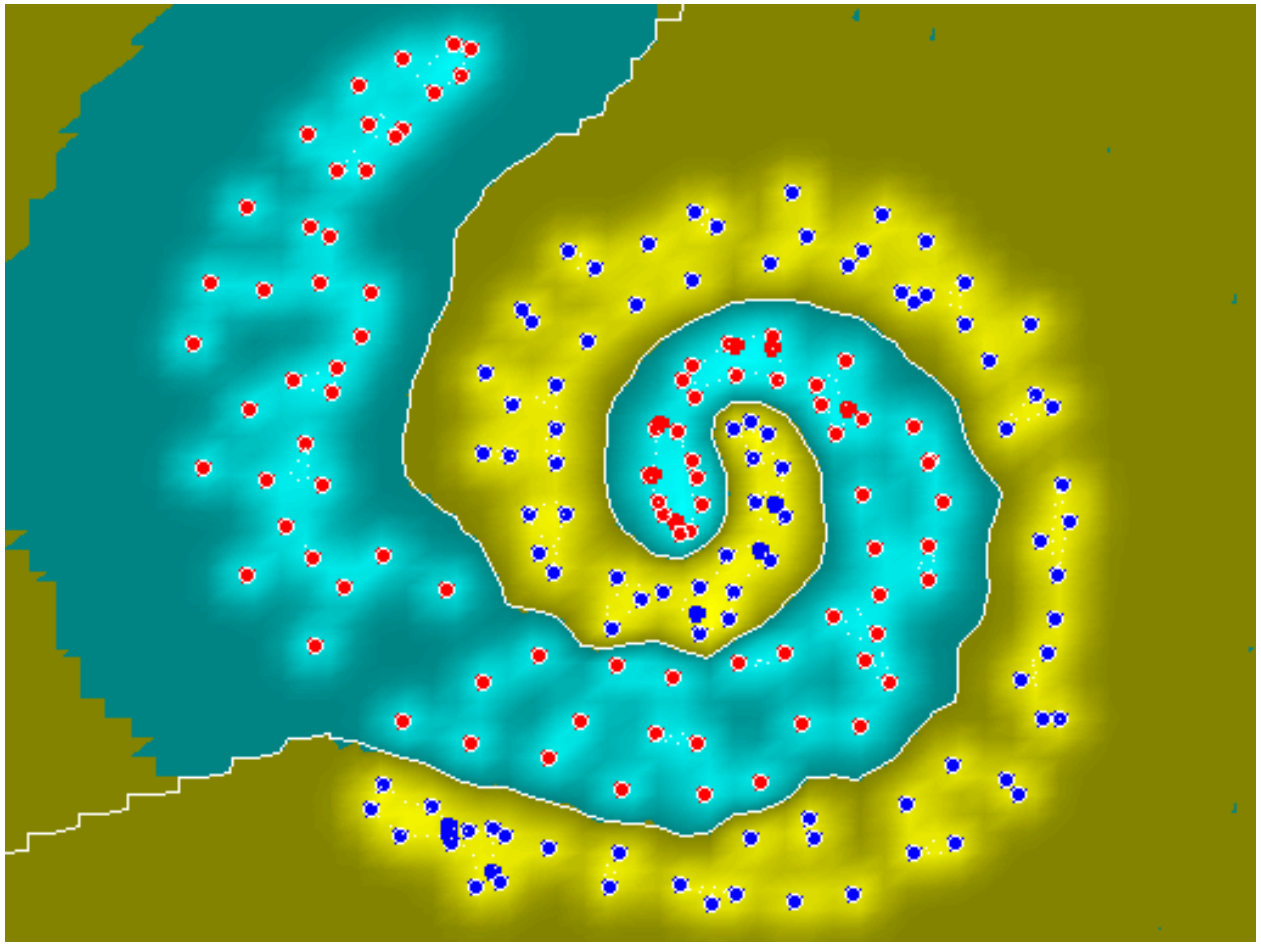
Gaussian kernel

$$\sigma = .25 \quad C = 1000$$



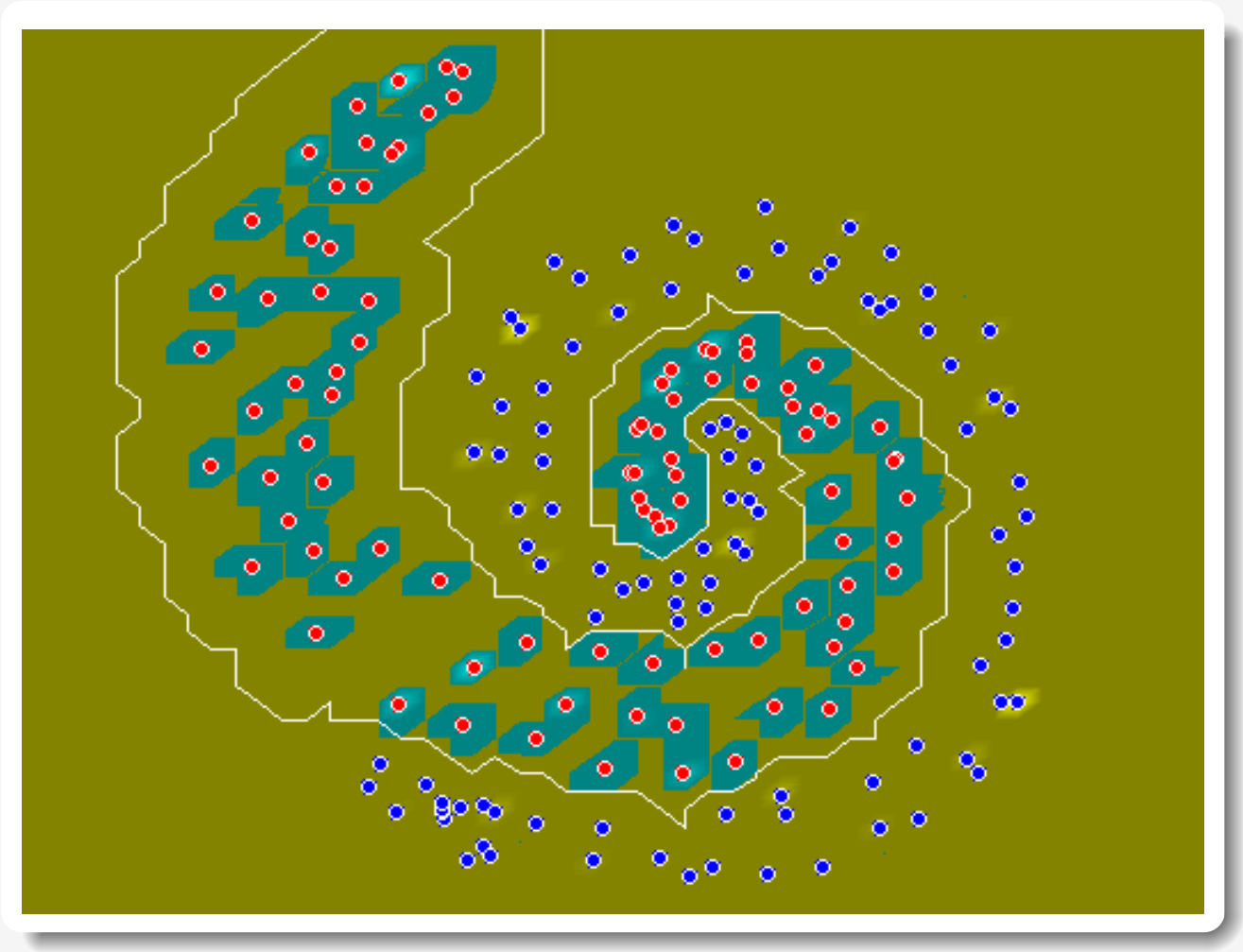
Gaussian kernel

$$\sigma = .05 \quad C = 1000$$



Gaussian kernel

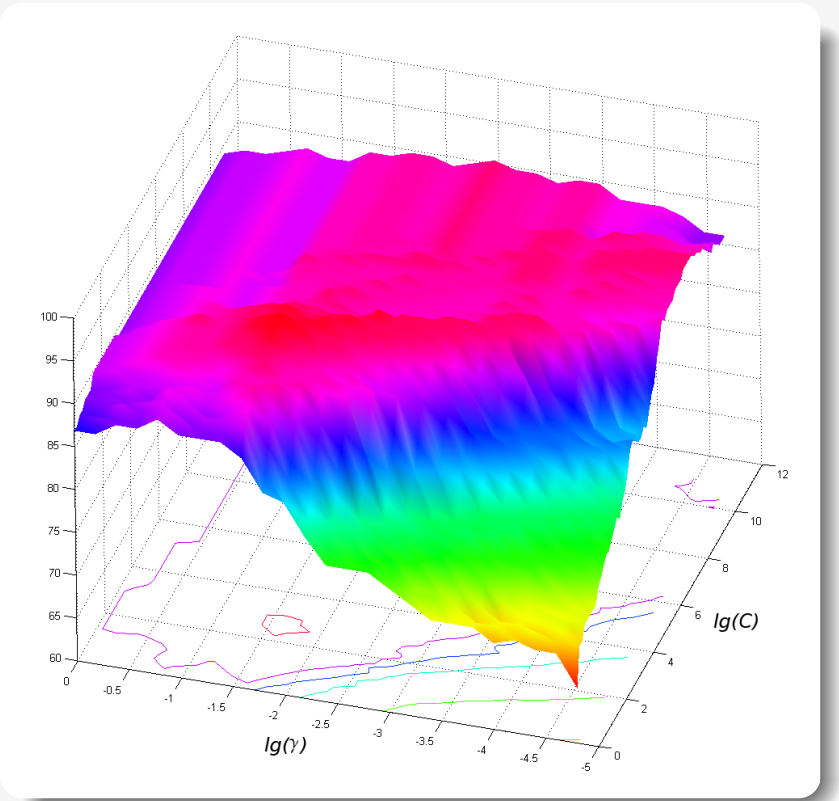
$$\sigma = .005 \quad C = 1000$$



Parameter Search

The Gaussian (=RBF) kernel is used in most cases, because it is very stable and has the advantage of requiring only two crucial parameters:

- λ is essentially the width of the Gaussian kernel function.
- C is the abovementioned regularization parameter controlling the amount of "slack".



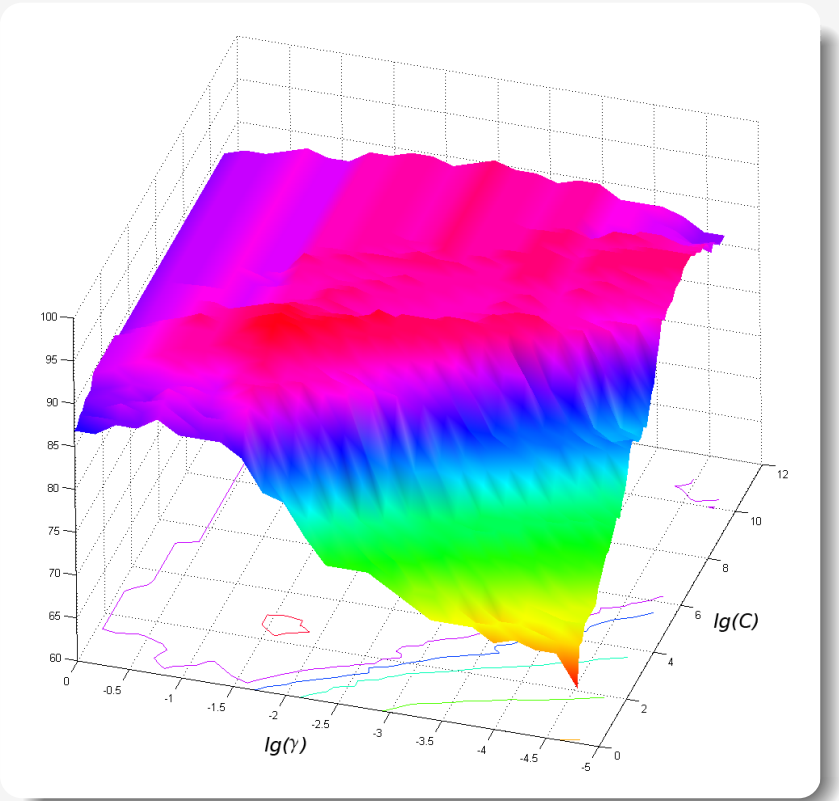
$\Rightarrow$ use an automated 2d grid search!

Classification performance at each grid point is evaluated using N-fold cross-validation

Parameter Search

The Gaussian (=RBF) kernel is used in most cases, because it is very stable and has the advantage of requiring only two crucial parameters:

- λ is essentially the width of the Gaussian kernel function.
- C is the abovementioned regularization parameter controlling the amount of "slack".

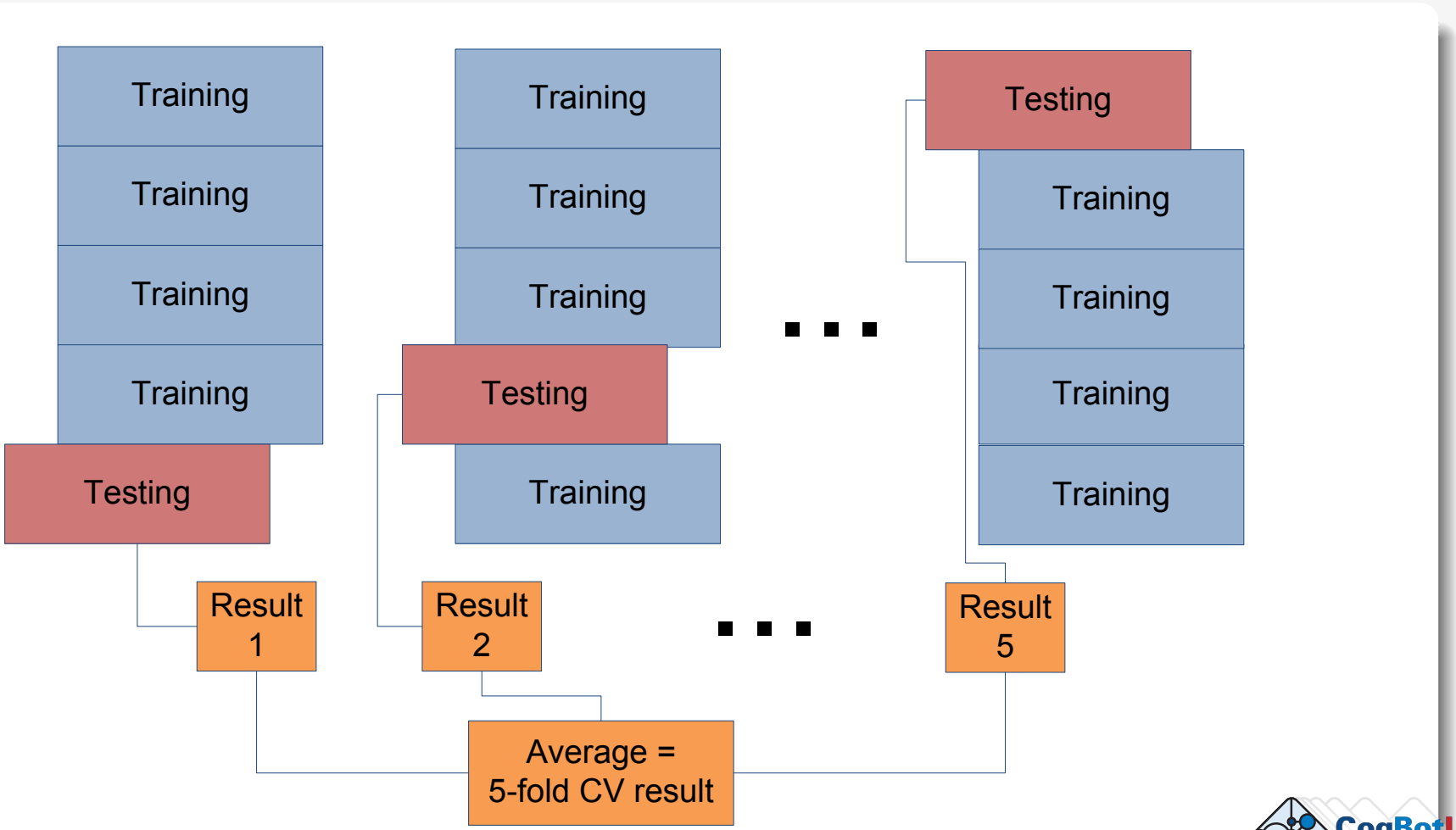


$\Rightarrow$ use an automated 2d grid search!

Classification performance at each grid point is evaluated using N-fold cross-validation

N-fold Cross-Validation

Split data into N parts, train on N-1 of them and use the N^{th} for testing (here $N = 5$). This is feasible due to quick and deterministic training of SVMs (but can be done for NNs, too, of course)



Multiclass SVMs

Problem: SVMs can only discern two classes!

Solution: Multiclass data sets need to be somehow broken down into binary problems.

Caveat: There is an ongoing discussion of how to best achieve this [**?**].

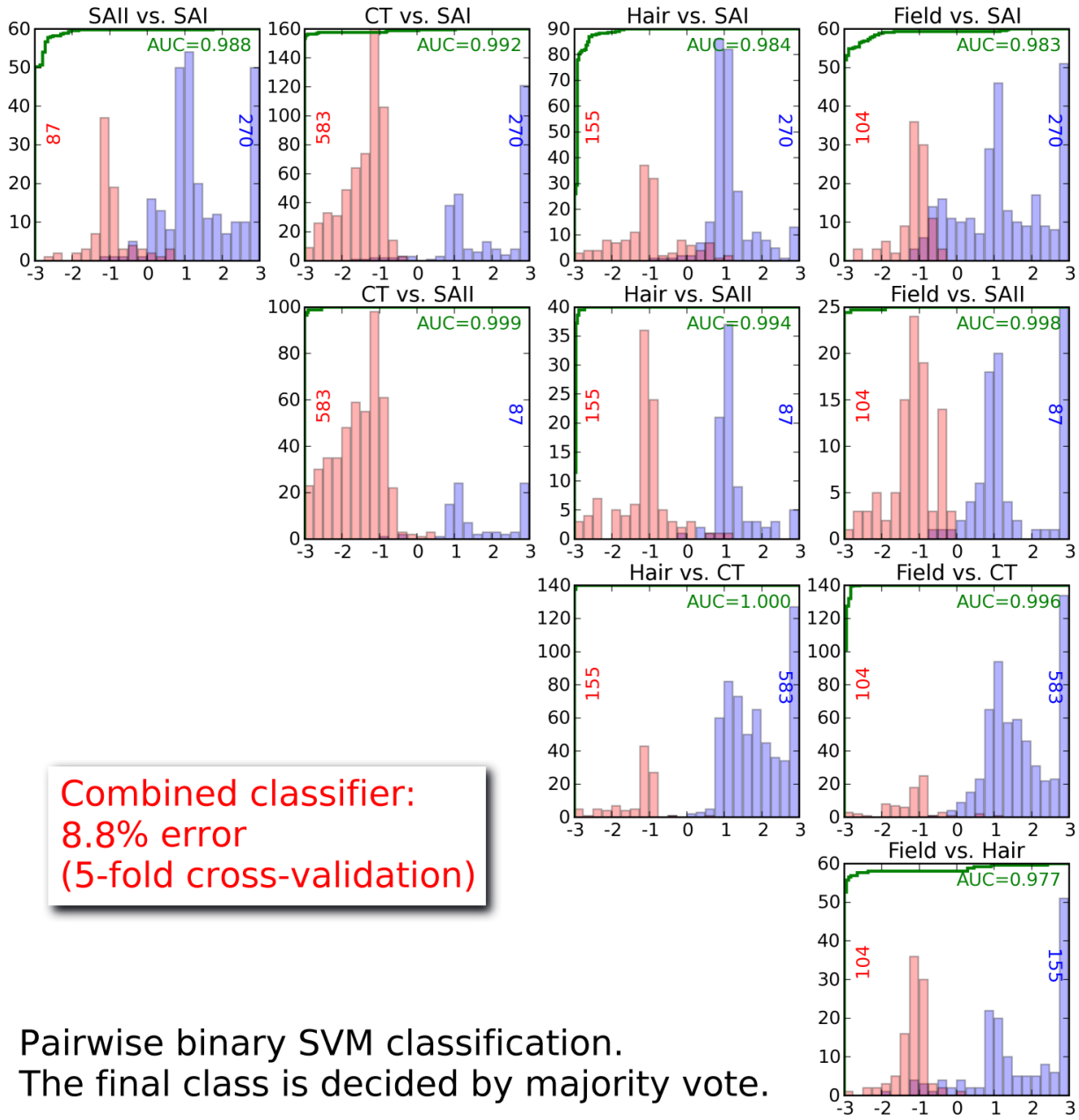
It is likely that we lose some optimality here. But there are two common and simple, but still relatively performant solutions:

one-vs-one: Split the data in to pairs of classes and train an SVM on each pair. When faced with unknown data, present it to all such SVMs and calculate the distances from the boundary, $d = \mathbf{w}^T\mathbf{x} + b$ for each one. Then use a voting mechanism to decide the class it is in.

- Pro: Possible to derive class membership probabilities from the raw distances [**?**].
- Con: With K classes, need to train $K(K-1)/2$ SVMs.

one-vs-rest: Separate one class from the rest of the data and train an SVM on this problem. Repeat for each class.

- Pro: Simplest solution.
- Con: Very unbalanced binary class distribution.



Combined classifier:
8.8% error
(5-fold cross-validation)

Pairwise binary SVM classification.
The final class is decided by majority vote.