Visual Computing Institute | RWTH AACHEN UNIVERSITY

Elements of Machine Learning & Data Science

Winter semester 2023/24

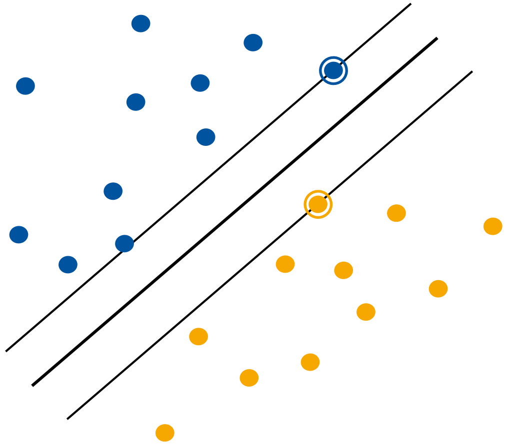
Lecture 17 – Support Vector Machines I

12.12.2023

Prof. Bastian Leibe

Machine Learning Topics

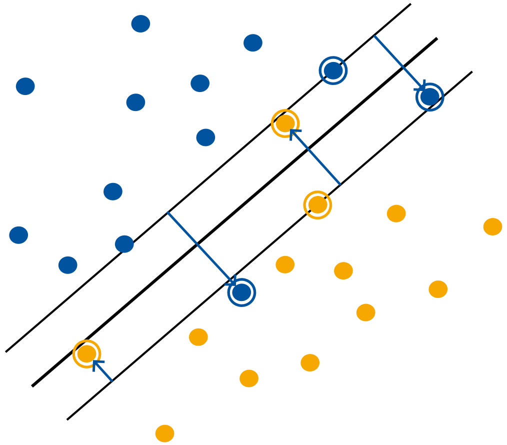
1. Introduction to ML
2. Probability Density Estimation
3. Linear Discriminants
4. Linear Regression
5. Logistic Regression
6. **Support Vector Machines**
7. (AdaBoost)
8. Neural Network Basics



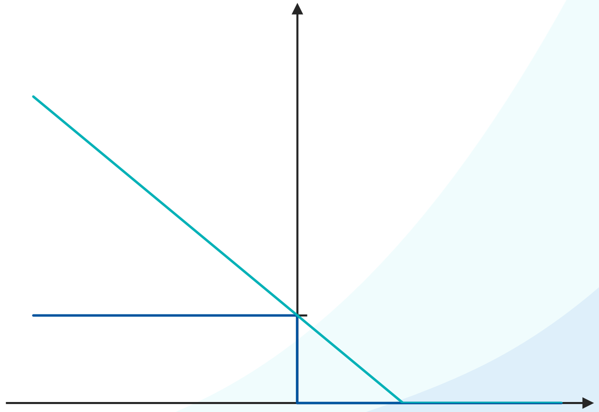
Maximum Margin Classification

$$L_p(\mathbf{w}, b, \mathbf{a})$$
$$L_d(\mathbf{a})$$

Primal & Dual Form



Soft-Margin SVM



Hinge Loss

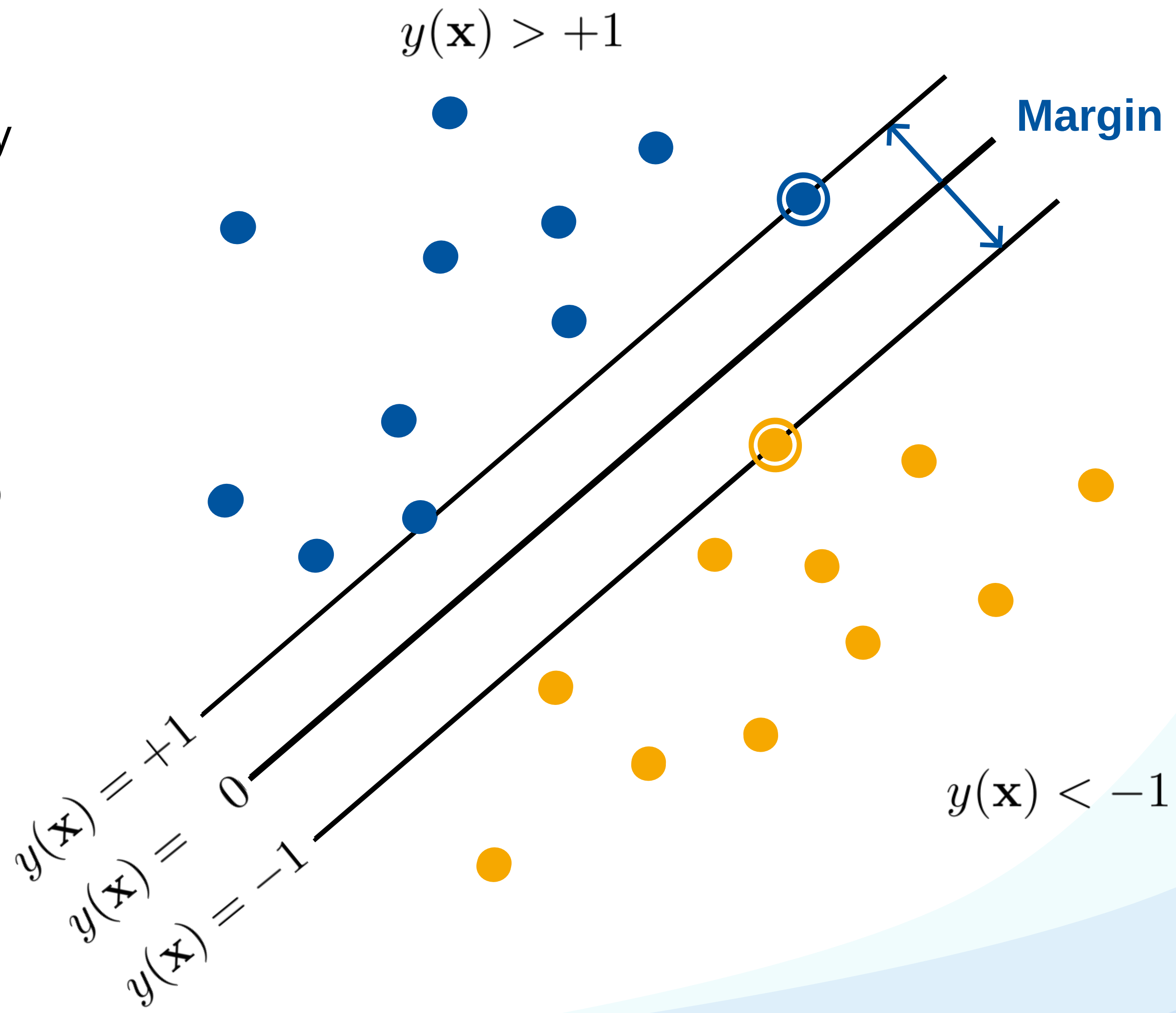
Recap: Maximum Margin Classification

- Intuitively, we want to choose the classifier which leaves maximal "safety room" for future data points.
- This classifier has the largest margin between positive and negative points.
- We can rescale $\mathbf{w}$ such that the distance of the points on the margin to the decision boundary is exactly 1.

$$t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) = 1$$

- If the data is linearly separable, then for all points, the following must hold:

$$t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) \geq 1 \quad \forall n$$



- Optimization problem
  - Find the hyperplane with maximum margin by optimizing:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

"Maximize the margin"

such that

$$t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) \geq 1 \quad \forall n$$

"such that each point is on the correct side of the margin"

- This is a quadratic programming problem with linear constraints.

Recap: Constrained Optimization with Lagrange Multipliers

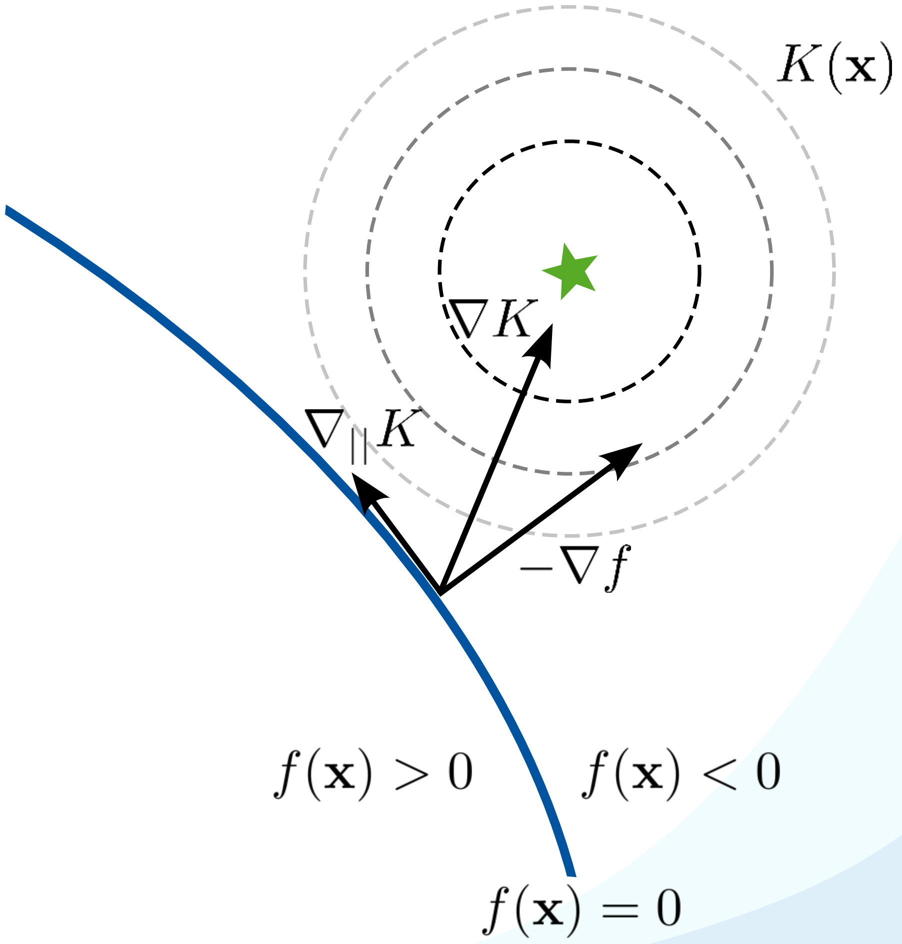
- If we want to minimize $K(\mathbf{x})$ subject to $f(\mathbf{x}) \geq 0$, we optimize the Lagrangian form

$$\mathcal{L}(\mathbf{x}, \lambda) = K(\mathbf{x}) - \lambda f(\mathbf{x})$$

  - *minimize* w.r.t. $\mathbf{x}$
  - *maximize* w.r.t. λ

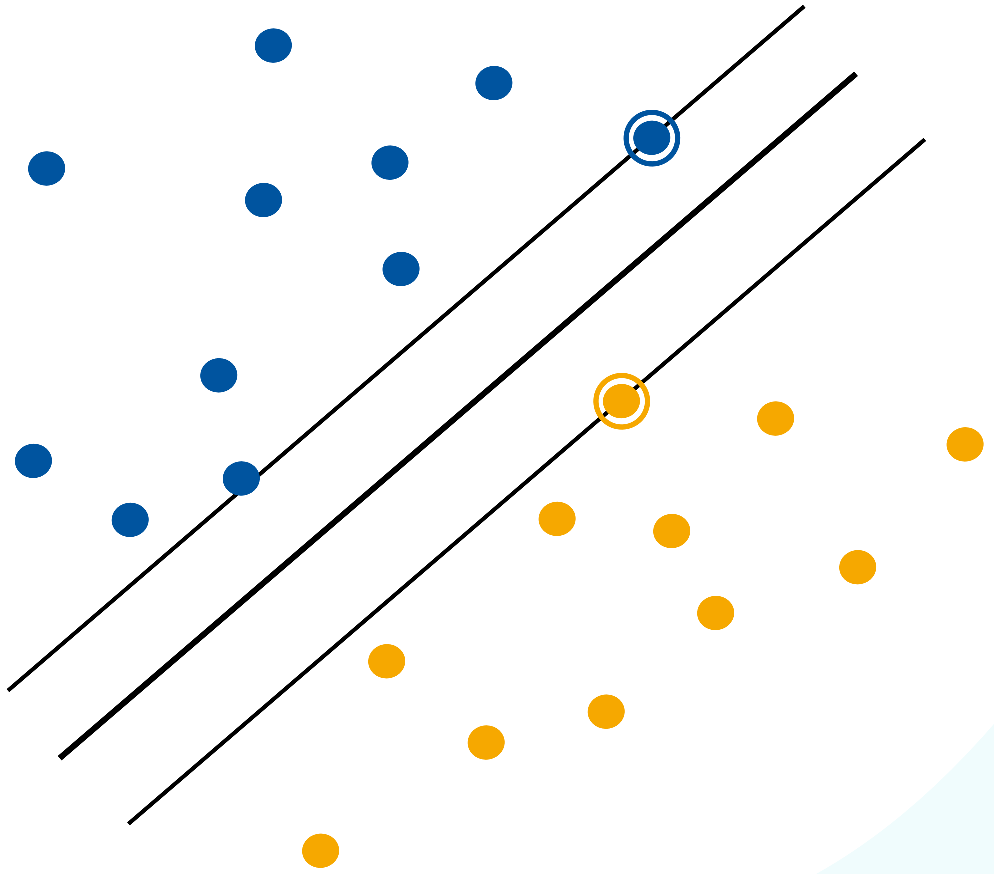
- I.e., we introduce an auxiliary variable λ for every constraint. λ is called a Lagrange multiplier.
- All valid solution need to fulfill the Karush-Kuhn-Tucker (KKT) conditions

$$\lambda \geq 0$$
$$f(\mathbf{x}) \geq 0$$
$$\lambda f(\mathbf{x}) = 0$$



Support Vector Machines

1. Maximum Margin Classification
2. **Primal Formulation**
3. Dual Formulation
4. Soft-Margin SVMs
5. Non-linear SVMs
6. Error Function Analysis



Primal SVM Formulation

- Recall the SVM objective:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2, \quad \text{such that} \quad t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) \geq 1 \quad \forall n$$

- We introduce positive Lagrange multipliers $a_n \geq 0$ and get the primal form of SVMs:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left[t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) - 1\right]$$

Necessary and sufficient conditions:

$$a_n \geq 0$$
$$t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) - 1 \geq 0$$
$$a_n[t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) - 1] = 0$$

KKT conditions:

$$\lambda \geq 0$$
$$f(\mathbf{x}) \geq 0$$
$$\lambda f(\mathbf{x}) = 0$$

Lagrangian Formulation

- We want to minimize the primal form:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left[t_n(\mathbf{w}^\top \mathbf{x}_n + b) - 1\right]$$

$$\frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial b} = \sum_{n=1}^{N} a_n t_n \qquad \frac{\partial L(\mathbf{w}, b, \mathbf{a})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$$

- Setting the gradients for $\mathbf{w}, b$ to zero, we get:

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \boxed{\sum_{n=1}^{N} a_n t_n = 0} \qquad \frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \boxed{\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n}$$

- The hyperplane is computed as a linear combination of training examples:

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$$

- Additionally, the solution needs to fulfill

$$a_n \left[t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) - 1 \right] = 0$$

- This implies $a_n > 0$ only for those points for which

$$\left[t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) - 1 \right] = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$$

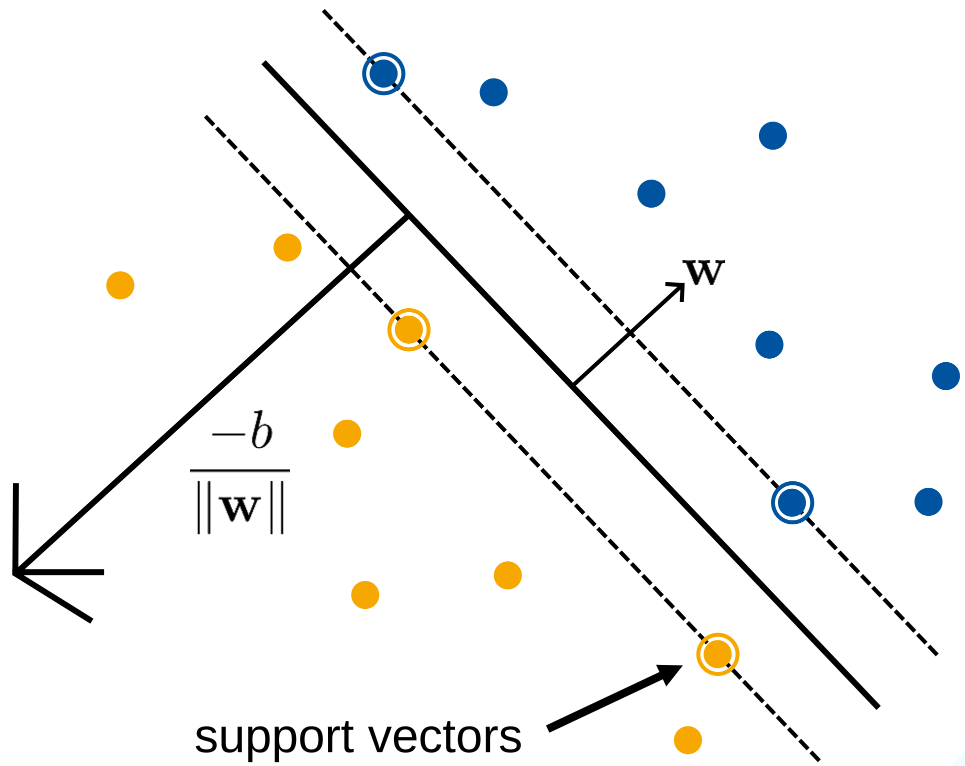
KKT conditions:

$$a_n \geq 0$$
$$t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) - 1 \geq 0$$
$$a_n[t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) - 1] = 0$$

Only some data points influence the decision boundary!

Intuition

- The training points with $a_n > 0$ are called support vectors.
- They are the points on the margin.
- This makes the SVM robust to “too correct” points!



- We still need to find b.
- Observation: Any support vector $\mathbf{x}_n$ satisfies

$$t_n y(\mathbf{x}_n) = t_n \left(\sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^\mathsf{T} \mathbf{x}_n + b \right) = 1$$

- Using $t_n^2 = 1$, we can derive

$$b = t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^\mathsf{T} \mathbf{x}_n$$

- In practice, it is more robust to average over all support vectors:

$$b = \frac{1}{N_\mathcal{S}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^\mathsf{T} \mathbf{x}_n \right)$$

Advantages

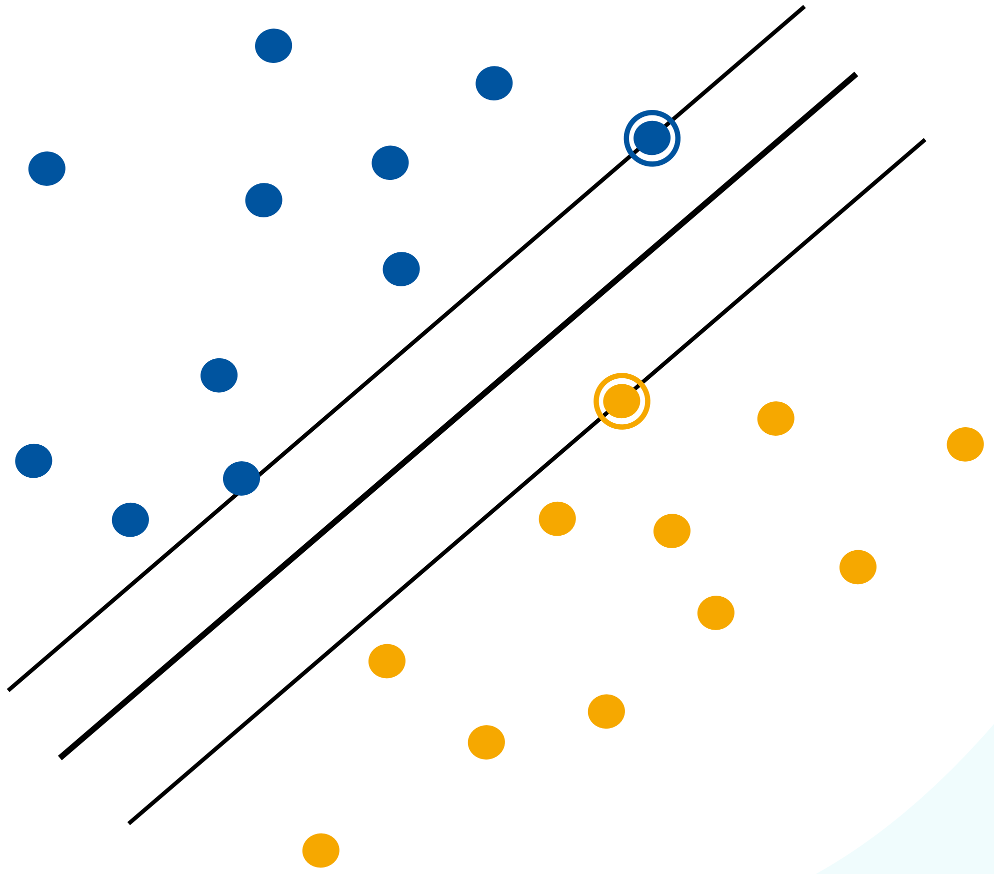
- SVMs yield a linear classifier with "guaranteed" generalization capability.
- Convex optimization, yields globally optimal solution.
- Solution depends only on a subset of the input data points, the support vectors.
- Automatic robustness against "too correct" data points.

Limitations

- Need to solve quadratic programming problem: time complexity for that is cubic in the number of variables.
- Here: Time complexity is in $\mathcal{O}(D^3)$.
- Scaling to high-dimensional data is difficult.

Support Vector Machines

1. Maximum Margin Classification
2. Primal Formulation
3. **Dual Formulation**
4. Soft-Margin SVMs
5. Non-linear SVMs
6. Error Function Analysis



Reminder: Primal SVM Formulation

- SVM objective:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2, \quad \text{such that} \quad t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) \geq 1 \quad \forall n$$

- This is a Quadratic Programming (QP) problem with linear inequality constraints.
  - In order to solve it, we have derived the Lagrangian primal form

$$L_p(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left[t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) - 1\right]$$

  - We are *minimizing* this objective with respect to $\mathbf{w}$ and b, and *maximizing* with respect to $\mathbf{a}$.

Solving a QP

- SVM objective:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2, \quad \text{such that} \quad t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) \geq 1 \quad \forall n$$

- Solving QPs is a well-understood problem
  - Typically done with the help of a QP solver.
  - Solving a QP in K variables can be done in runtime $\mathcal{O}(K^3)$.

- In our case: $\mathbf{x}, \mathbf{w} \in \mathbb{R}^D$
  - #Variables: $D + 1$

  $\Rightarrow$ Complexity: $\mathcal{O}(D^3)$

Solving a QP

- SVM objective:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2, \quad \text{such that} \quad t_n(\mathbf{w}^\mathsf{T}\boldsymbol{\phi}(\mathbf{x}_n) + b) \geq 1 \quad \forall n$$

- Solving QPs is a well-understood problem
  - Typically done with the help of a QP solver.
  - Solving a QP in K variables can be done in runtime $\mathcal{O}(K^3)$.

- In our case: $\mathbf{x}, \mathbf{w} \in \mathbb{R}^D$
  - #Variables: $D + 1$

  $\Rightarrow$ Complexity: $\mathcal{O}(D^3)$

- With basis functions: $\boldsymbol{\phi}(\mathbf{x}), \mathbf{w} \in \mathbb{R}^M, \ M \gg D$
  - #Variables: $M + 1$

  $\Rightarrow$ Complexity: $\mathcal{O}(M^3)$

$\Rightarrow$ Curse of dimensionality, the SVM Primal Form does not scale well!

Dual Form of the SVM Objective

- Maximize

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^\mathsf{T}\mathbf{x}_n)$$

under the conditions

$$a_n \geq 0 \quad \forall n$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

- We now have an optimization problem in N variables.

$\Rightarrow$ Complexity: $\mathcal{O}(N^3)$

For the derivation, please watch the video

Discussion

- What have we gained?
  - Previous complexity was $\mathcal{O}(D^3)$, now it is $\mathcal{O}(N^3)$.
  - *Isn't this much worse for large training sets???*

- However, the dual form has several advantages
  1. SVMs have sparse solutions: $a_n \neq 0$ only for support vectors.
     – This makes very efficient algorithms possible.
     – E.g., Sequential Minimal Optimization (SMO)
     – Effective runtime between $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$.

  2. No dependency on the dimensionality anymore.
     – We can work with high-dimensional feature spaces!

Advantages

- Optimization problem only depends on the Lagrange multipliers a_n resulting in a worst-case runtime complexity of $\mathcal{O}(N^3)$.
- Since SVMs have sparse solutions and only few $a_n \neq 0$, specialized algorithms can solve the dual form very efficiently.
- The complexity of QP optimization no longer depends on the dimensionality of the feature space. This makes it possible to use very high-dimensional feature spaces.

Limitations

- Evaluating the SVM decision function

$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$$

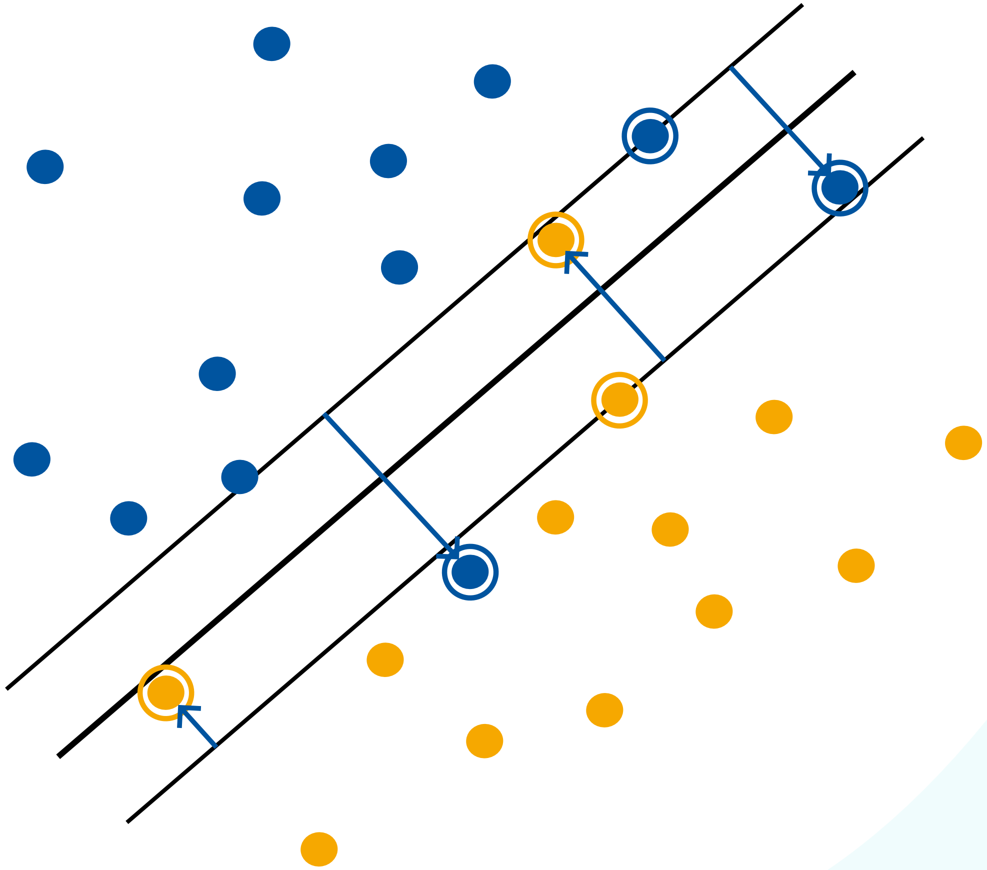
with

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)$$

is still costly for high-dimensional feature spaces $\phi(\mathbf{x})$.

Support Vector Machines

1. Maximum Margin Classification
2. Primal Formulation
3. Dual Formulation
4. **Soft-Margin SVMs**
5. Non-linear SVMs
6. Error Function Analysis

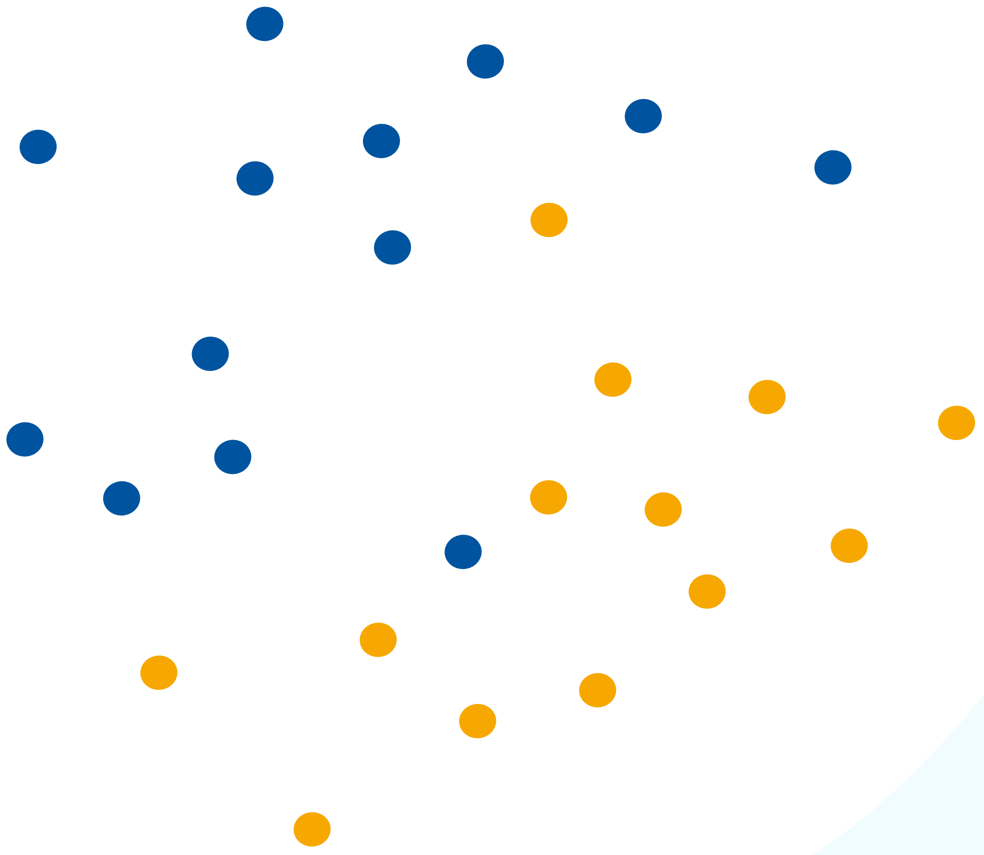


Soft-Margin SVM

- So far, we assumed linearly separable data.
  - Our current formulation has no solution if the data are not linearly separable!

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2,$$

such that $\quad t_n(\mathbf{w}^\mathsf{T}\mathbf{x}_n + b) \geq 1 \quad \forall n$



- Need to introduce tolerance to outlier data points.
  - The resulting model is called soft-margin SVM.

Slack Variables

- For non-linearly separable data, not all constraints can be satisfied:

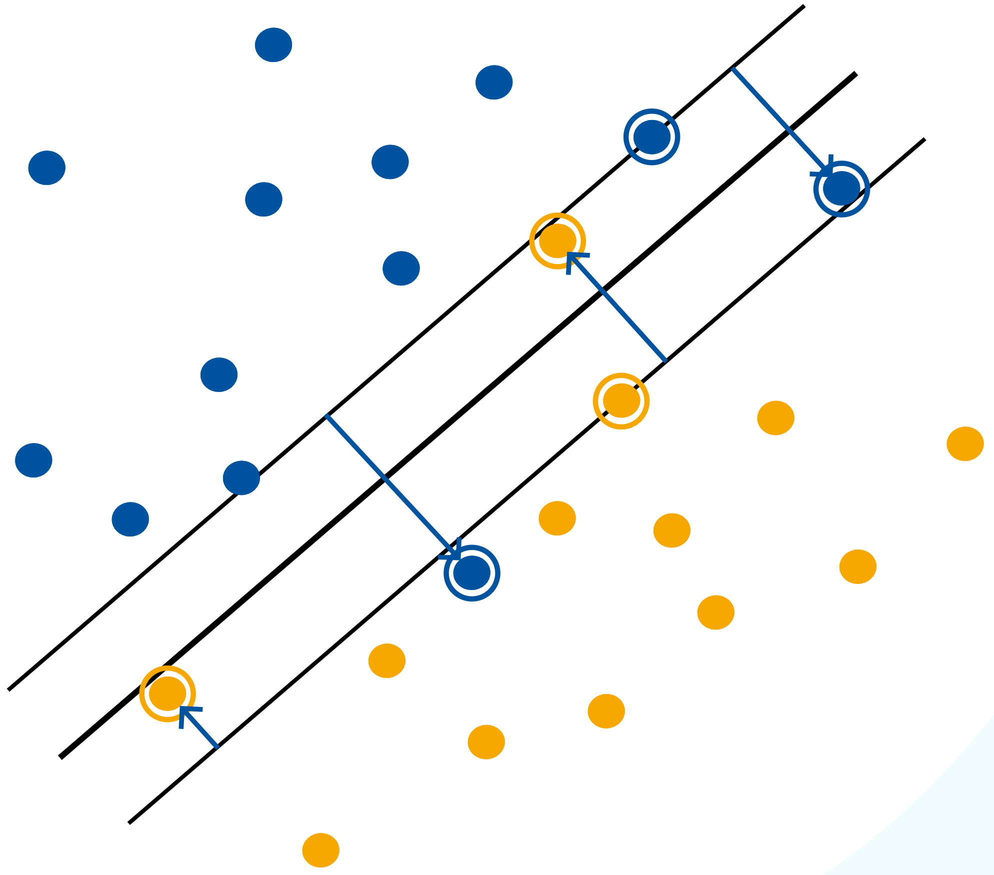
$$\mathbf{w}^\top \mathbf{x}_n + b \geq +1 \quad \text{for } t_n = +1$$
$$\mathbf{w}^\top \mathbf{x}_n + b \leq -1 \quad \text{for } t_n = -1$$

- Idea: Introduce slack variables $\xi_n \geq 0$:

$$\mathbf{w}^\top \mathbf{x}_n + b \geq +1 - \xi_n \quad \text{for } t_n = +1$$
$$\mathbf{w}^\top \mathbf{x}_n + b \leq -1 + \xi_n \quad \text{for } t_n = -1$$

$\Rightarrow$ We allow some datapoints to violate the constraint.

- For those points, the slack ξ_n makes up for the difference.



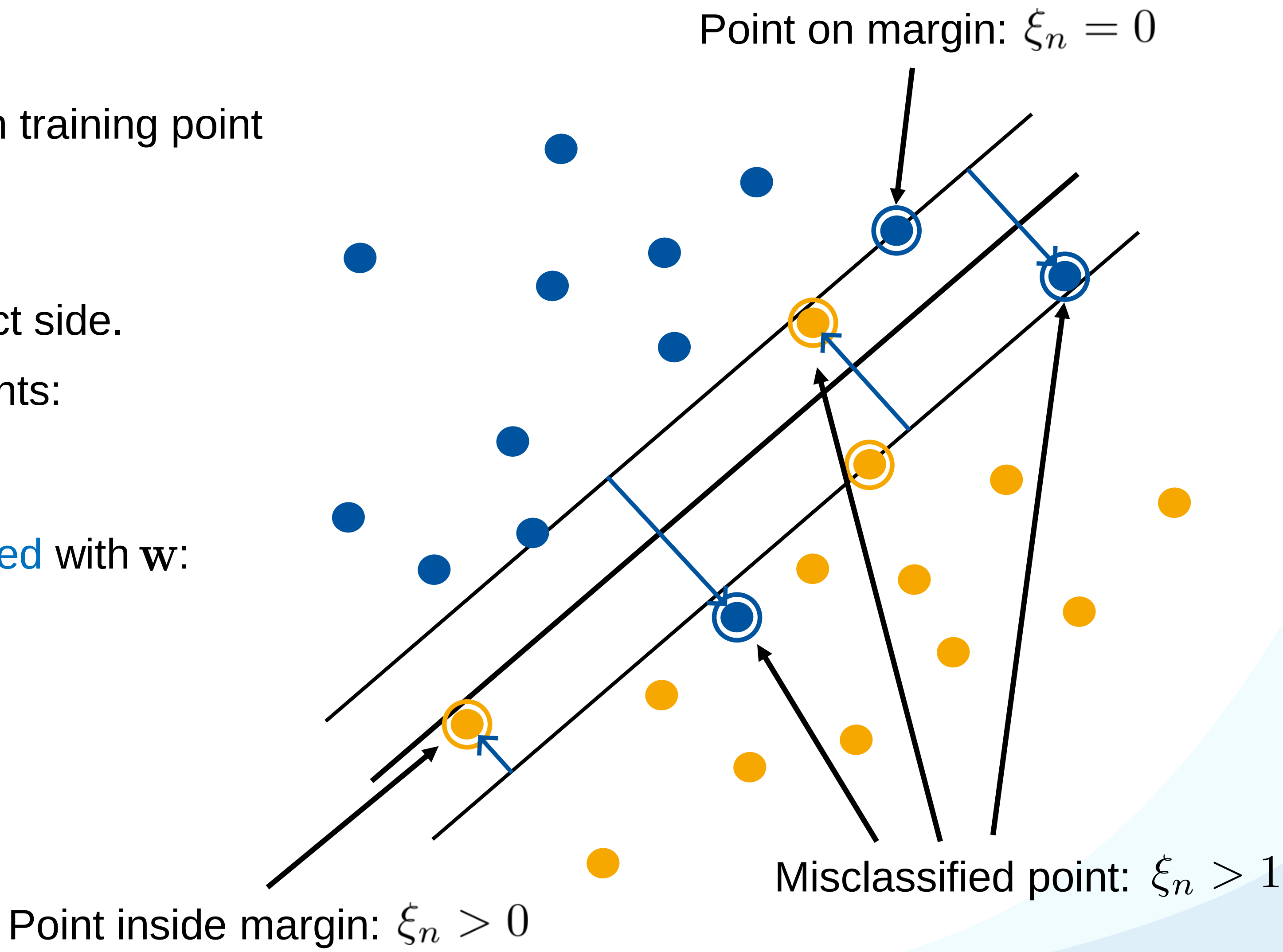
- Slack variables
  - One slack variable ξ_n for each training point

- Effect
  - $\xi_n = 0$ for points on the correct side.
  - Linear penalty for all other points: $\xi_n = |t_n - y(\mathbf{x}_n)|$

- Slack variables are jointly optimized with $\mathbf{w}$:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$

where C is a tradeoff parameter.



New Primal Formulation

- Minimize

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \underbrace{\sum_{n=1}^{N} a_n\left[t_n(y(\mathbf{x}_n) - 1 + \xi_n\right]}_{\text{Constraint}} - \underbrace{\sum_{n=1}^{N}\mu_n\xi_n}_{\text{Constraint}}$$

Constraint $t_n y(\mathbf{x}_n) \geq 1 - \xi_n$ Constraint $\xi_n \geq 0$

- KKT conditions

$$\begin{aligned} a_n &\geq 0 & \mu_n &\geq 0 \\ t_n y(\mathbf{x}_n) - 1 + \xi_n &\geq 0 & \xi_n &\geq 0 \\ a_n\left[t_n y(\mathbf{x}_n) - 1 + \xi_n\right] &= 0 & \mu_n\xi_n &= 0 \end{aligned}$$

New Dual Formulation

- Maximize

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m (\mathbf{x}_m^\mathsf{T} \mathbf{x}_n)$$

- Under the side conditions

$$0 \leq a_n \leq C \quad \forall n$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

This is the only difference to before.

New Solution

- The decision hyperplane is again a linear combination of training samples:

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \mathbf{x}_n$$

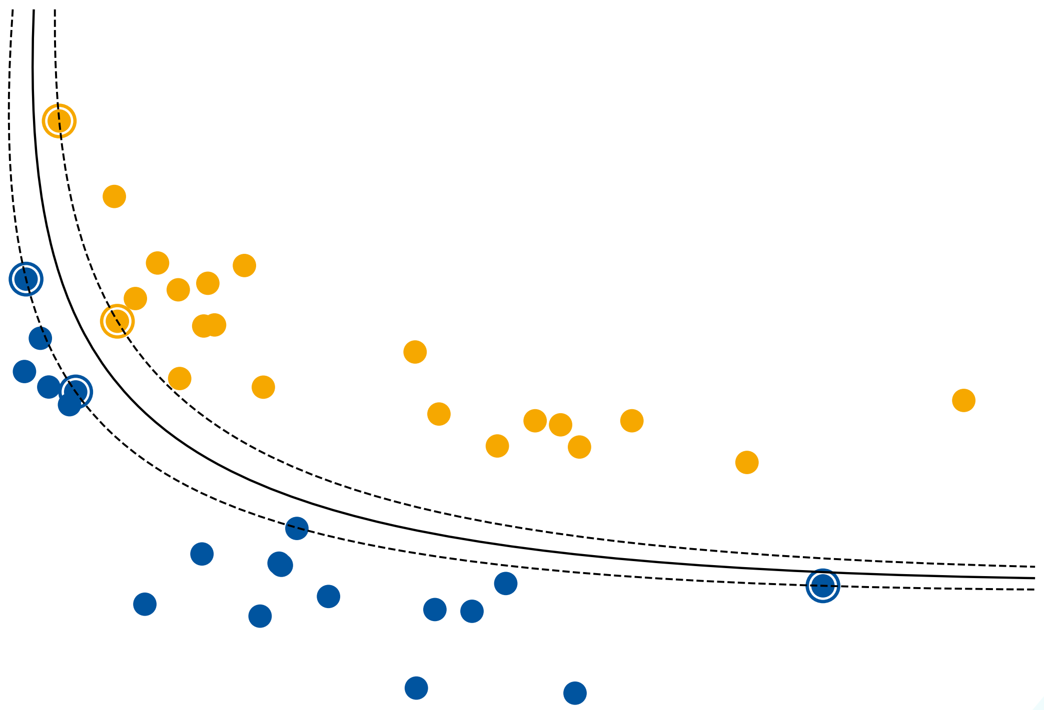
- This is still a sparse solution:
  - $a_n = 0$ for points on the correct side of the margin
  - Slack points with $\xi_n > 0$ are now also support vectors!

- Compute b by averaging over support vectors (points with $0 < a_n < C$):

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{n \in \mathcal{S}} \left(t_n - \sum_{m \in \mathcal{S}} a_m t_m \mathbf{x}_m^{\mathsf{T}} \mathbf{x}_n \right)$$

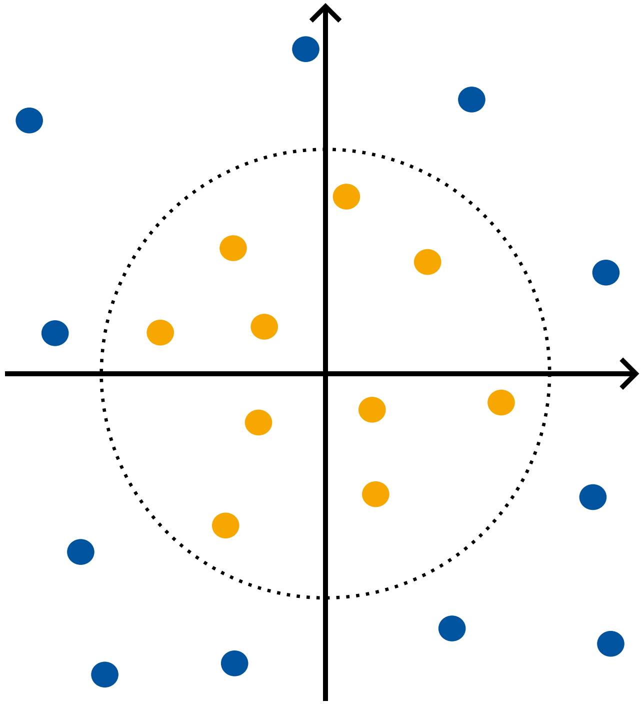
Support Vector Machines

1. Maximum Margin Classification
2. Primal Formulation
3. Dual Formulation
4. Soft-Margin SVMs
5. **Non-Linear SVMs**
6. Error Function Analysis



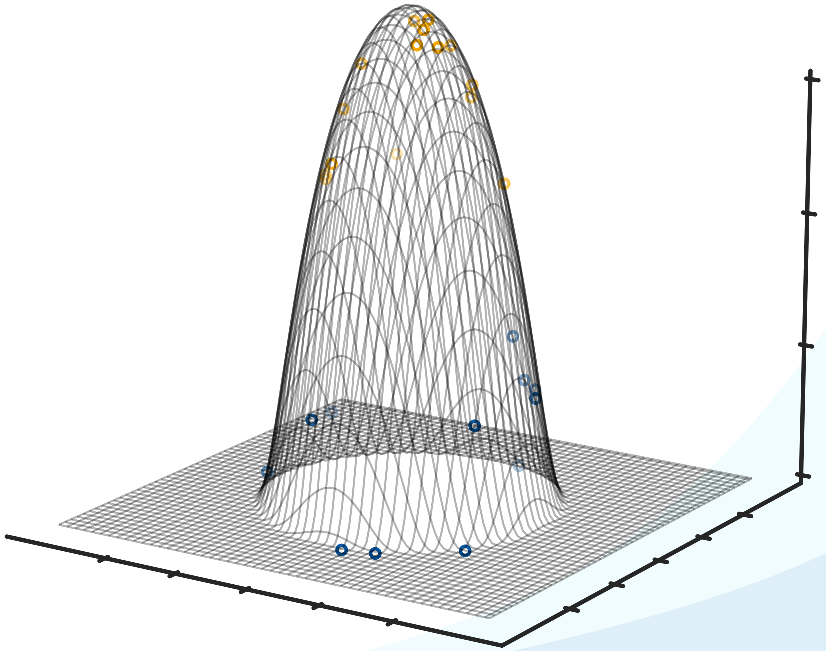
Non-Linear SVMs

- So far, we have only considered linear decision boundaries.
- We now combine non-linear basis functions with SVMs.



$$\phi : \mathbb{R}^D \rightarrow \mathbb{R}^M$$

$$M \gg D$$



Feature Spaces

- We have already seen non-linear basis functions:
  - Apply a nonlinear transformation ϕ to the data points $\mathbf{x}_n$:

    $$\mathbf{x} \in \mathbb{R}^D, \quad \boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^M$$

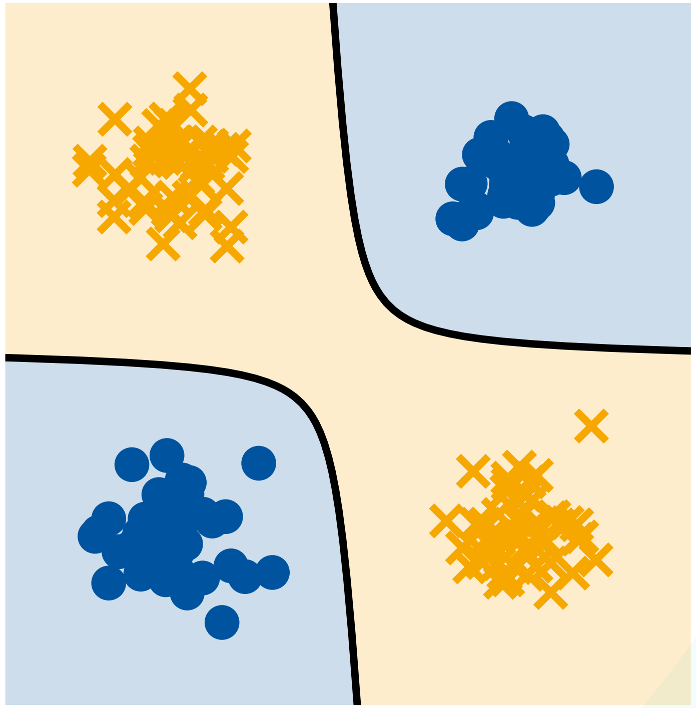
  - Classify with a hyperplane in higher-dim. space $\mathbb{R}^M$:

    $$\mathbf{w}^\mathsf{T}\phi(\mathbf{x}) + b = 0$$

  $\Rightarrow$ Linear classifier in $\mathbb{R}^M$, nonlinear classifier in $\mathbb{R}^D$.

- Let us now apply this to SVMs…
  - We can train our SVM on the transformed features $\phi(\mathbf{x})$ to get non-linear decision boundaries.
  - Usually, $M \gg D$: evaluating $\mathbf{w}^\mathsf{T}\phi(\mathbf{x})$ can be quite expensive!

$\mathbf{x} = (x_1, x_2)^\mathsf{T}$



$\phi(\mathbf{x}) = (x_1, x_2, x_1x_2, x_1^2, x_2^2)^\mathsf{T}$

The Kernel Trick

- On a closer look, $\phi(\mathbf{x})$ only appears in the form of dot products:

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m (\boldsymbol{\phi}(\mathbf{x}_m)^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{x}_n))$$

$$y(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}) + b$$

$$= \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)^{\mathsf{T}}\boldsymbol{\phi}(\mathbf{x}) + b$$

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n)$$

The Kernel Trick

- On a closer look, $\phi(\mathbf{x})$ only appears in the form of dot products:

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m (\phi(\mathbf{x}_m)^\top \phi(\mathbf{x}_n))$$

$$y(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$$

$$= \sum_{n=1}^{N} a_n t_n \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}) + b$$

$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$$

Define a kernel function $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$

$\Rightarrow$ *Use the kernel instead of the dot product.*

The Kernel Trick

- On a closer look, $\phi(\mathbf{x})$ only appears in the form of dot products:

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_m, \mathbf{x}_n)$$

$$y(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\phi(\mathbf{x}) + b$$

$$= \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

$$k(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^{\mathsf{T}}\phi(\mathbf{y})$$

Define a kernel function $k(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^{\mathsf{T}}\phi(\mathbf{y})$

$\Rightarrow$ *Use the kernel instead of the dot product.*

- $k(\cdot,\cdot)$ implicitly maps the data to some higher-dimensional space, without having to compute $\phi(\mathbf{x})$.

When Can We Apply the Kernel Trick?

- In order for this to work, $k(\cdot,\cdot)$ needs to define an implicit mapping.
- Formally
  - A function $k(\mathbf{x}_1, \mathbf{x}_2) : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is a kernel function, iff
    - There is a mapping $\phi(\mathbf{x}) : \mathbb{R}^D \to \mathcal{H}$ such that

$$k(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^{\mathsf{T}} \phi(\mathbf{x}_2) \quad \forall \mathbf{x}_1, \mathbf{x}_2$$

- *When will this be the case?*

- When is a function $k(\mathbf{x}_1, \mathbf{x}_2)$ a valid kernel function? Two ways to check:

  1. Every Gram matrix K of k is symmetric positive definite:

$$K = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

A matrix M is positive definite if all eigenvalues of K are positive.

    - This is easy to verify for a given training set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$.
    - Unfortunately, it has to hold for *every possible* such set.

$\Rightarrow$ *Very hard to prove in practice.*

- When is a function $k(\mathbf{x}_1, \mathbf{x}_2)$ a valid kernel function? Two ways to check:

2. We can construct valid kernels from other valid kernels:
   - Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following combinations will also be valid

$$k(\mathbf{x}, \mathbf{x}') = c \cdot k_1(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) k_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = polynomial(k_1(\mathbf{x}, \mathbf{x}')) \quad (with\ nonnegative\ coefficients)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}'))$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') k_2(\mathbf{x}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\boldsymbol{\phi}(x), \boldsymbol{\phi}(x'))$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathsf{T}} \mathbf{A} \mathbf{x}'$$

$\Rightarrow$ *Much easier to apply in practice.*

New SVM Formulation

- Maximize

$$L_d(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_m, \mathbf{x}_n)$$

under the constraints $0 \leq a_n \leq C \quad \forall n$

$$\sum_{n=1}^{N} a_n t_n = 0$$

- Classify new data points using

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

Example: Polynomial Kernel

- We slightly adjust the polynomial basis function that we know:

$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2)^\mathsf{T}$$

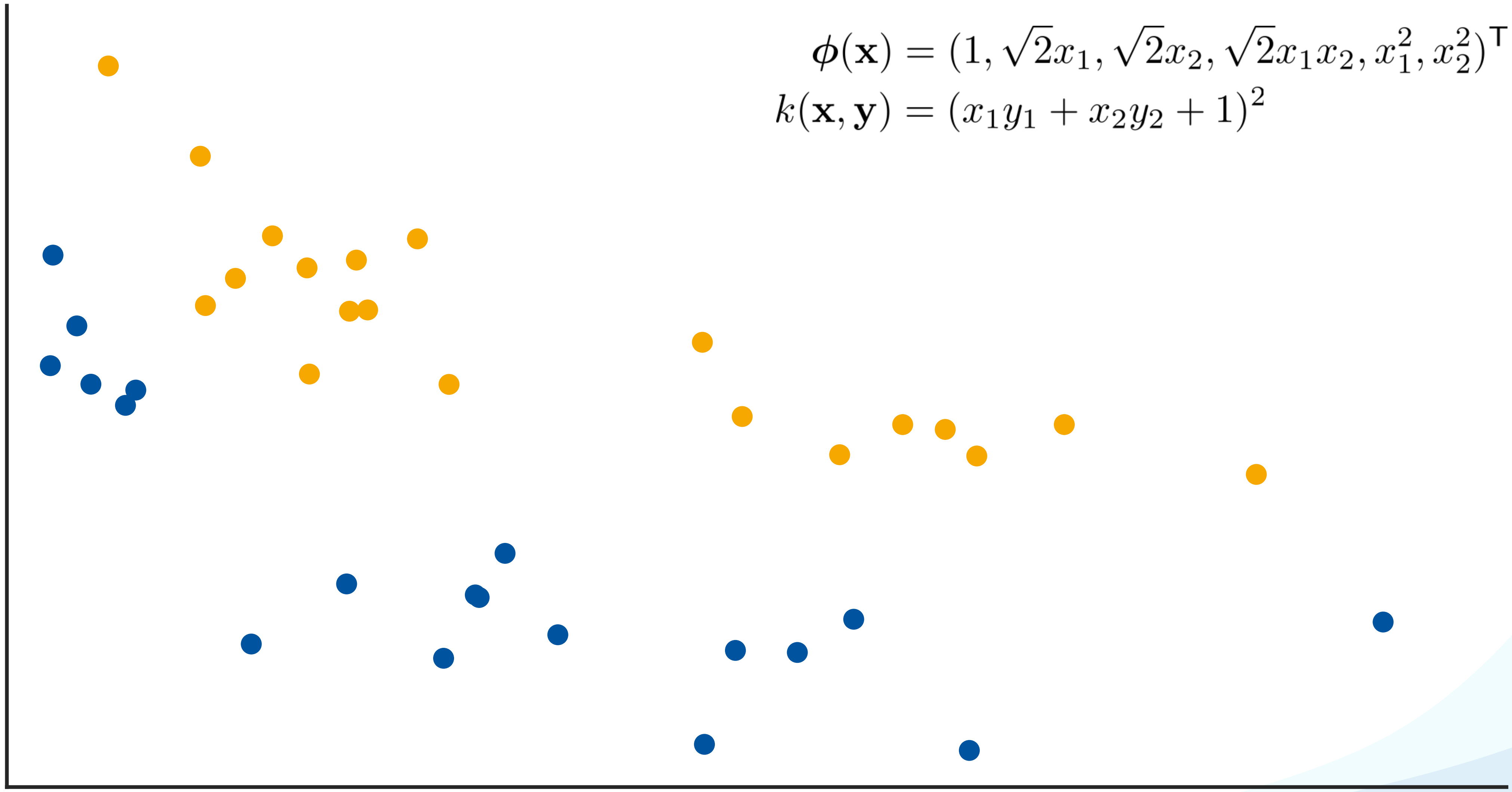
$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\mathsf{T}\mathbf{y} + 1)^2 = \phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{y})$$

- In fact, $(\mathbf{x}^\mathsf{T}\mathbf{y} + 1)^p$ is the kernel function for a polynomial of degree p.

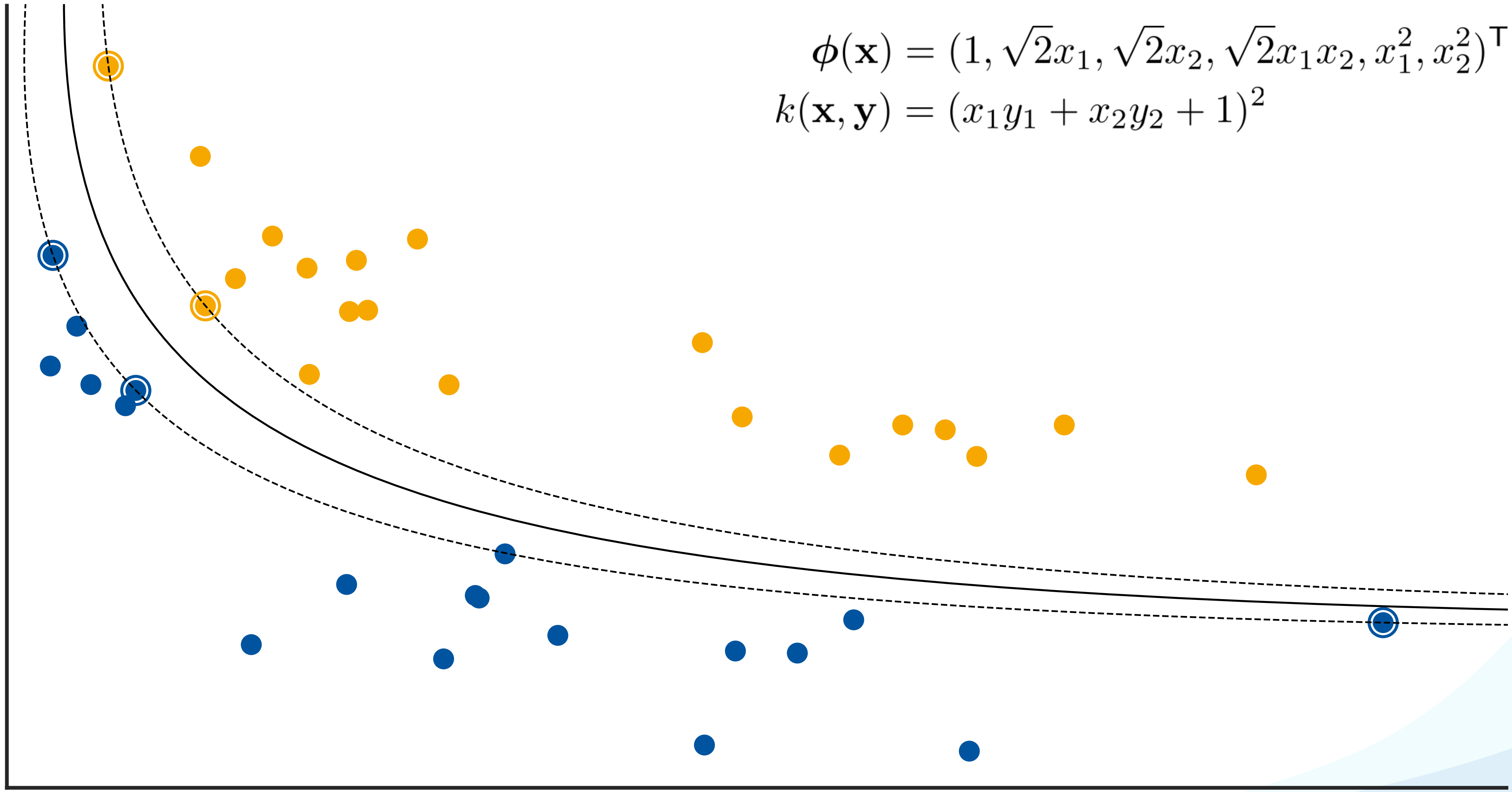
Example

$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2)^\mathsf{T}$$

$$k(\mathbf{x}, \mathbf{y}) = (x_1y_1 + x_2y_2 + 1)^2$$



Example



$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2)^\mathsf{T}$$
$$k(\mathbf{x}, \mathbf{y}) = (x_1y_1 + x_2y_2 + 1)^2$$

Advantages

- We can use high-dimensional or even infinite dimensional feature spaces
  - Since $\phi(\mathbf{x})$ is never computed explicitly.
- We can work with non-vector space data
  - We can define kernel functions for arbitrary data types!
  - Graphs, Sets, Sequences, Histograms, …
- Simple to use and work very well in most cases

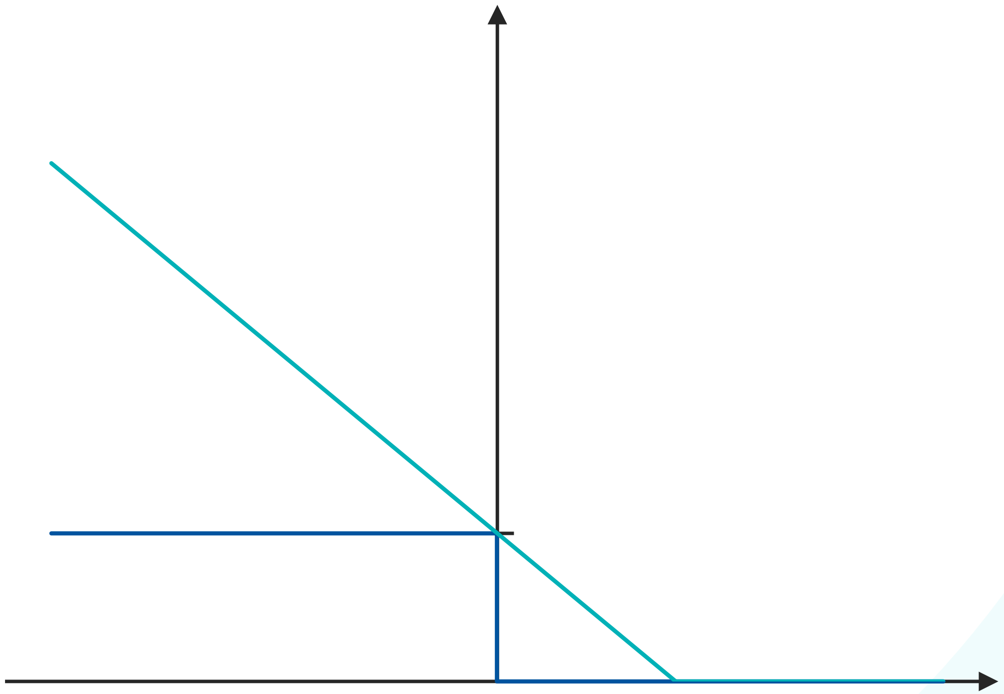
Limitations

- Which kernel to choose?
  - Model selection problem
- How to choose kernel parameters?
  - Hyperparameter optimization problem, usually solved by performing a grid search over the validation set
- Evaluation speed scales linearly with number of support vectors

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}_n, \mathbf{x}) + b$$

Support Vector Machines

1. Maximum Margin Classification
2. Primal Formulation
3. Dual Formulation
4. Soft-Margin SVMs
5. Non-linear SVMs
6. **Error Function Analysis**



Error Function Analysis

- We know how to formulate and optimize an SVM as a convex optimization problem:

$$\underset{\mathbf{w},\, b,\, \xi_n \in \mathbb{R}^+}{\arg\min} \; \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n$$

subject to the constraints

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

Error Function Analysis

- We know how to formulate and optimize an SVM as a convex optimization problem:

$$\underset{\mathbf{w},\, b,\, \xi_n \in \mathbb{R}^+}{\arg\min} \; \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{n=1}^{N} \xi_n$$

subject to the constraints

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n$$

But what error function does this correspond to?

- Integrate the constraints into the objective function:
  - Rewrite as $\xi_n \geq 1 - t_n y(\mathbf{x}_n)$
  - Thus, we obtain

$$\min_{\mathbf{w},\, b} E(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{n=1}^{N} [1 - t_n y(\mathbf{x}_n)]_+$$

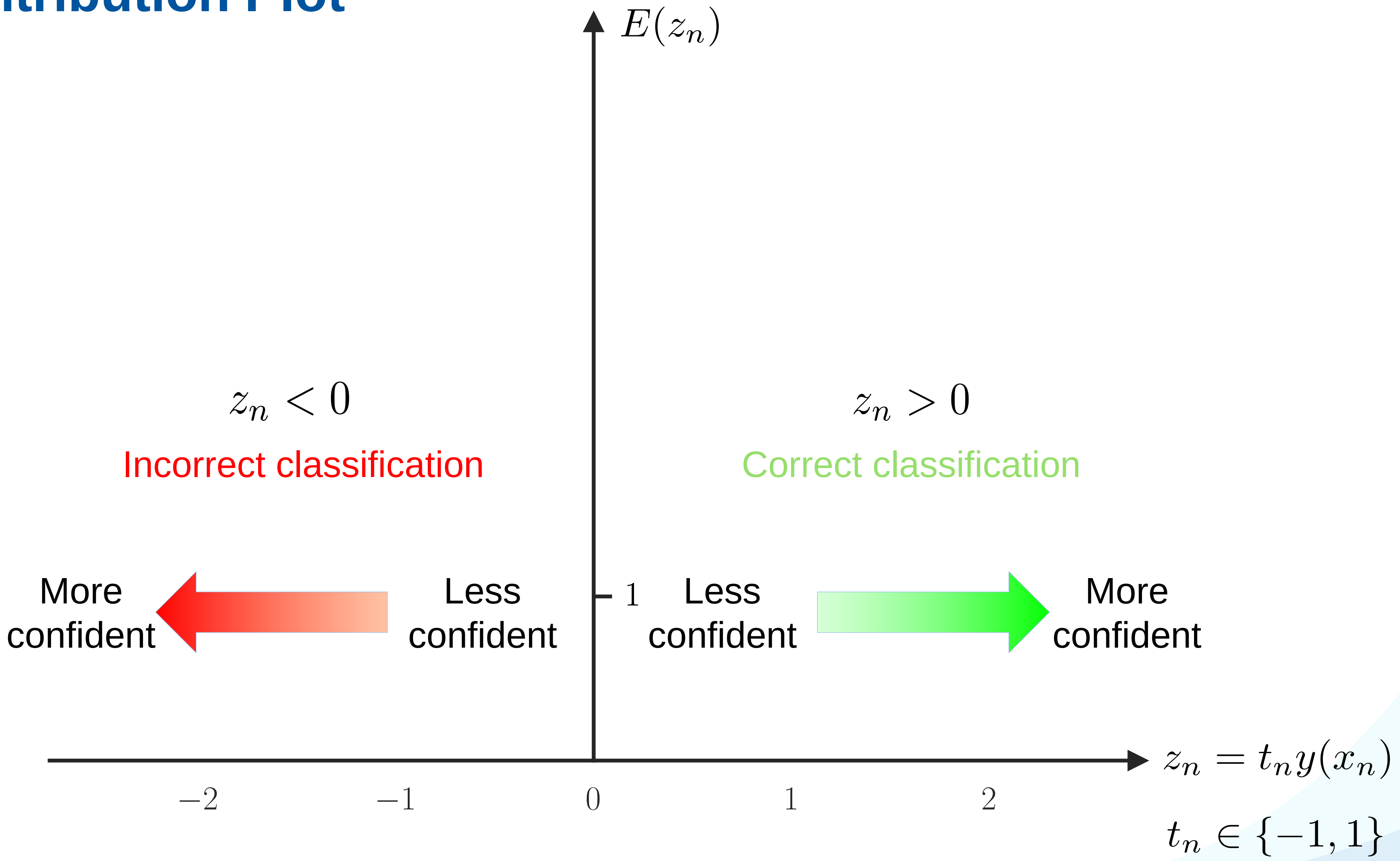
$$[x]_+ \equiv \max\{x, 0\}$$

The Hinge Loss

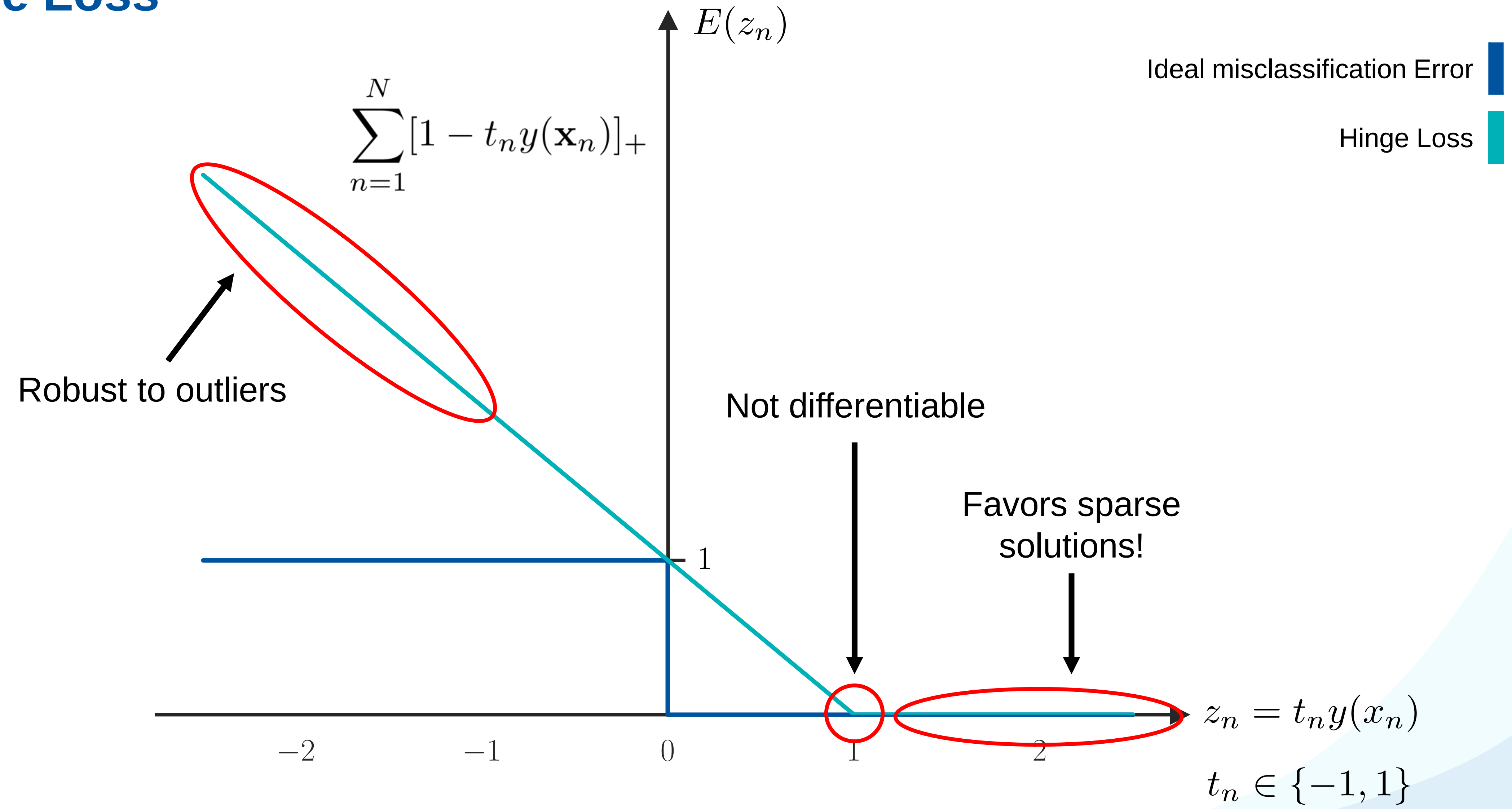
$$E(\mathbf{w}) = \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{L_2 \text{ regularization}} + C \underbrace{\sum_{n=1}^{N} [1 - t_n y(\mathbf{x}_n)]_+}_{\text{Hinge loss}}$$

- Regularization bounds parameter size.
- Hinge Loss enforces sparsity:
  - Only a subset of training data points actually influences the decision boundary.
  - Still, all input dimensions are used.
- This formulation corresponds to an unconstrained optimization of a non-differentiable function.
  - Very efficient: stochastic (sub-)gradient descent.

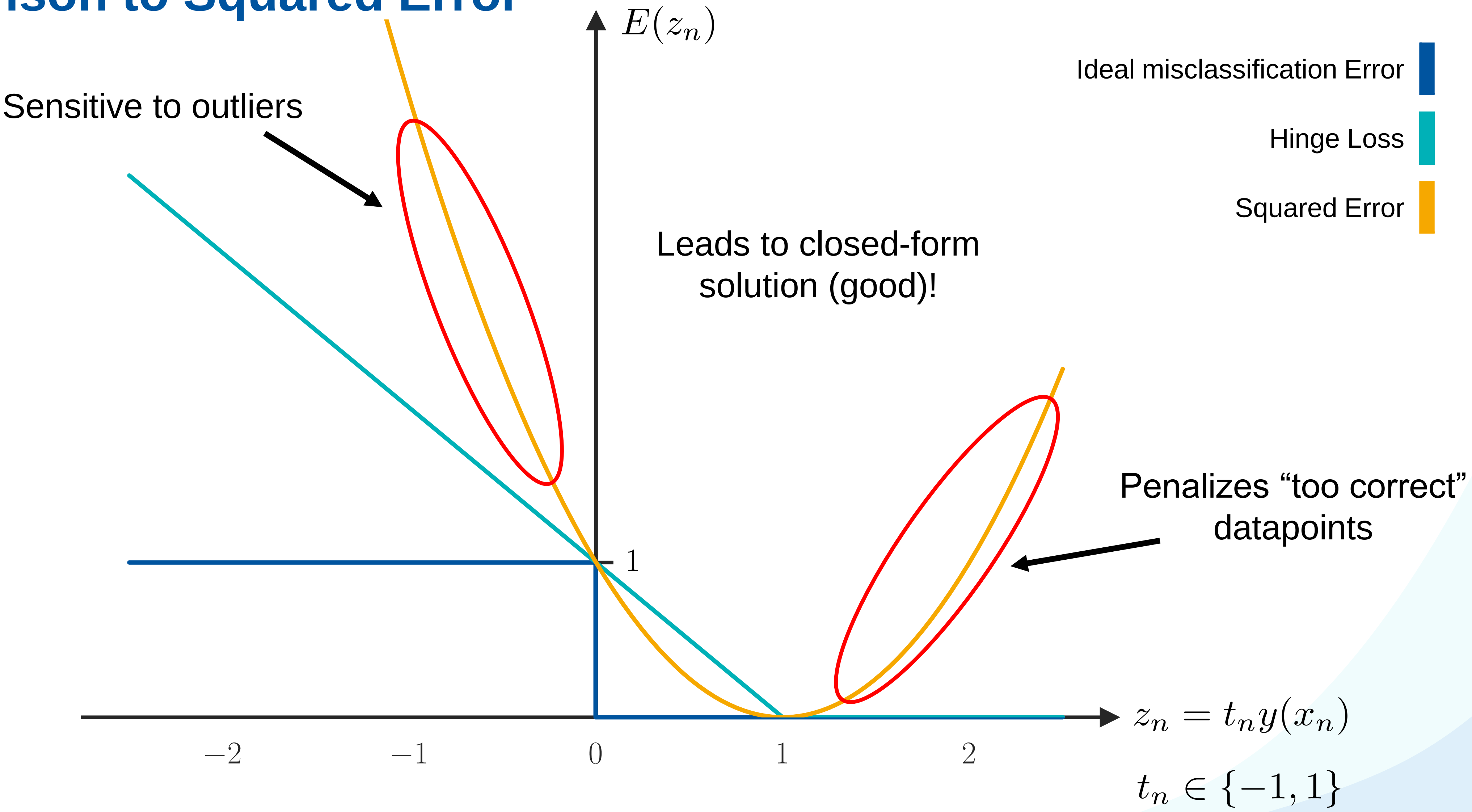
Error Contribution Plot



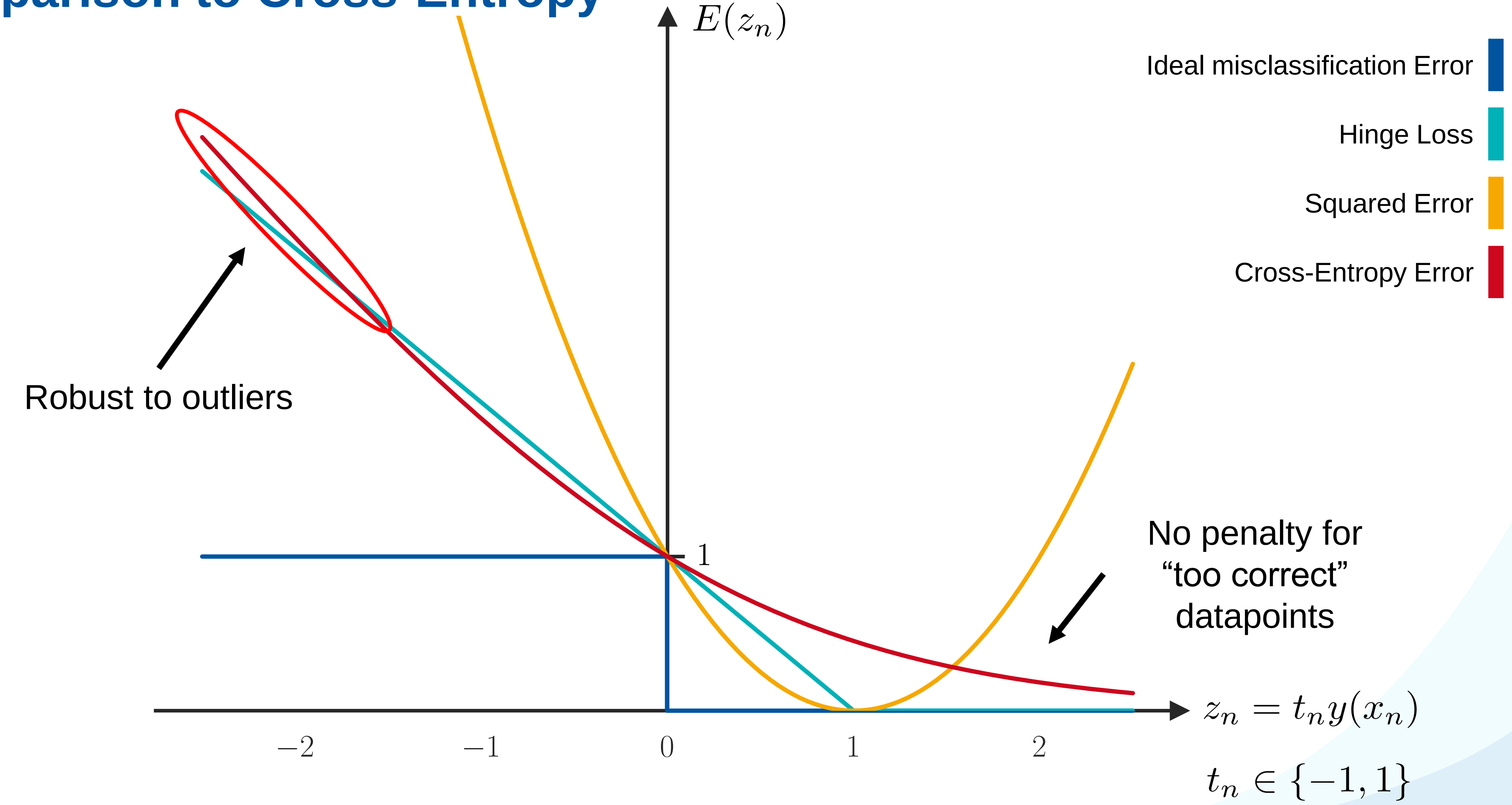
Hinge Loss


$E(z_n)$
Ideal misclassification Error
Hinge Loss
$\sum_{n=1}^{N} [1 - t_n y(\mathbf{x}_n)]_+$
Robust to outliers
Not differentiable
Favors sparse solutions!
1
-2
-1
0
1
2
$z_n = t_n y(x_n)$
$t_n \in \{-1, 1\}$

Comparison to Squared Error



Comparison to Cross-Entropy



Discussion: Hinge Loss

Advantages

- Favors sparse solutions that only depend on a subset of training data points.
- Robust to outliers (only a linear penalty for misclassified points).
- Convex function, unique minimum exists.

Limitations

- Not differentiable (cannot minimize this loss using standard gradient descent, but need to use subgradient descent).

References and Further Reading

- More information about SVMs is available in Chapter 7.1 of Bishop's book.

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006