Support Vector Machines

Jan Bronec

December 4-5, 2025

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

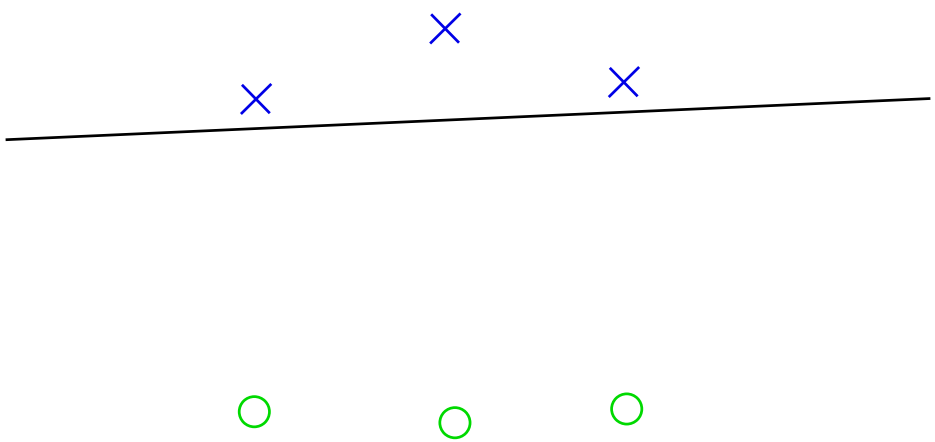
Revisiting linear classifiers

Perceptron

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

- We find $\boldsymbol{w}, b$ with the perceptron learning algorithm.

> It doesn't guarantee anything about the decision boundary.



Recall the simple linear classifiers

Perceptron

$$f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

- We find $\boldsymbol{w}, b$ with the Perceptron algorithm.

> It doesn't guarantee anything about the decision boundary.

> It will not converge for linearly non-separable data.

Logistic regression

$$p(\text{Positive} \mid \boldsymbol{x}; \boldsymbol{w}, b) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

- We find $\boldsymbol{w}, b$ by optimizing for NLL with SGD:

$$\boldsymbol{w}^*, b^* = \operatorname*{argmin}_{\boldsymbol{w}, b} \frac{1}{N} \sum_i -\log(p(C_i \mid \boldsymbol{x}_i; \boldsymbol{w}, b))$$

- It can handle even non-separable data
- We can help with separability using polynomial features.

> Polynomial features increase the complexity of learning and inference.

> Decision boundary may still be suboptimal.

Logistic regression

> Polynomial features increase the complexity of learning and inference.

Consider logistic regression with linear, quadratic, and cubic features computed by feature mapping $\varphi : \mathbb{R}^D \to \mathbb{R}^{1+D+D^2+D^3}$:

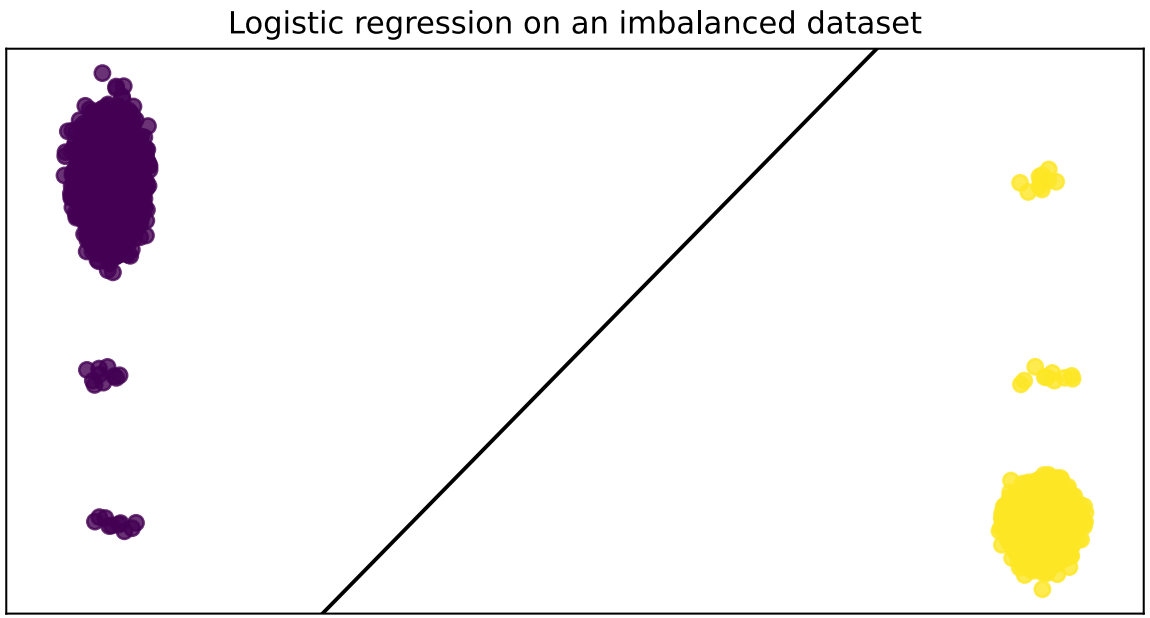
$$\varphi(\boldsymbol{x}) = \begin{pmatrix} 1 \\ x_i \\ \dots \\ x_i x_j \\ \dots \\ x_i x_j x_k \\ \dots \end{pmatrix} \text{ with } i, j, k \in \{1 \dots D\}$$

For input of dimensionality D, one step of SGD takes $\mathcal{O}(D^3)$ per example.

Logistic regression

> Decision boundary may still be suboptimal.

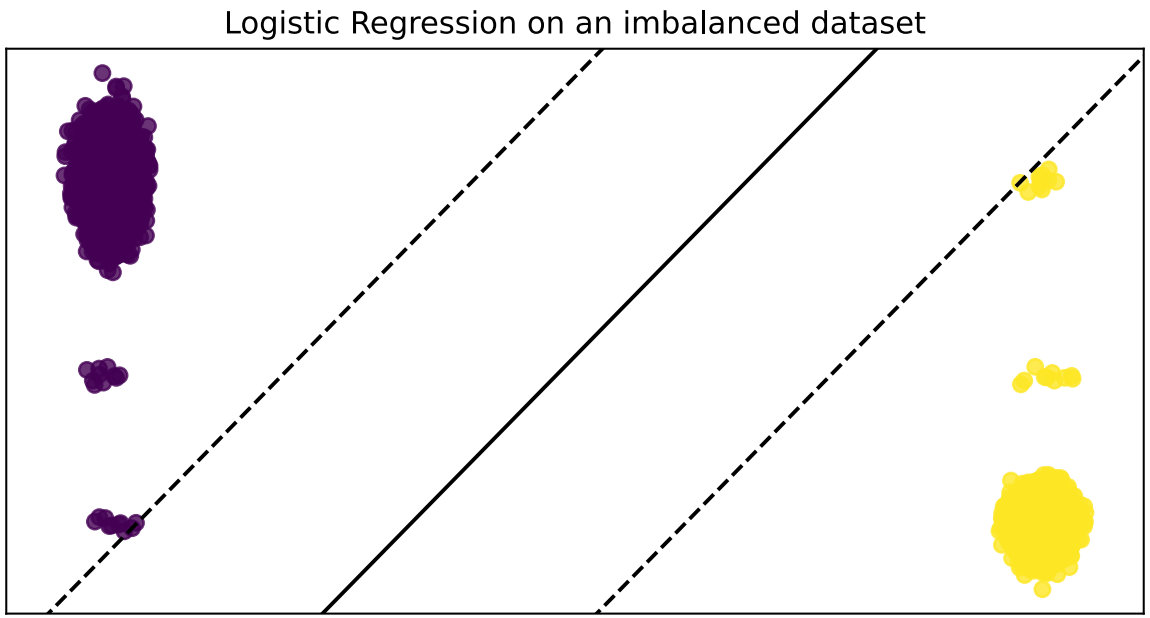
The decision boundary is easily skewed by an imbalanced dataset.



Logistic regression

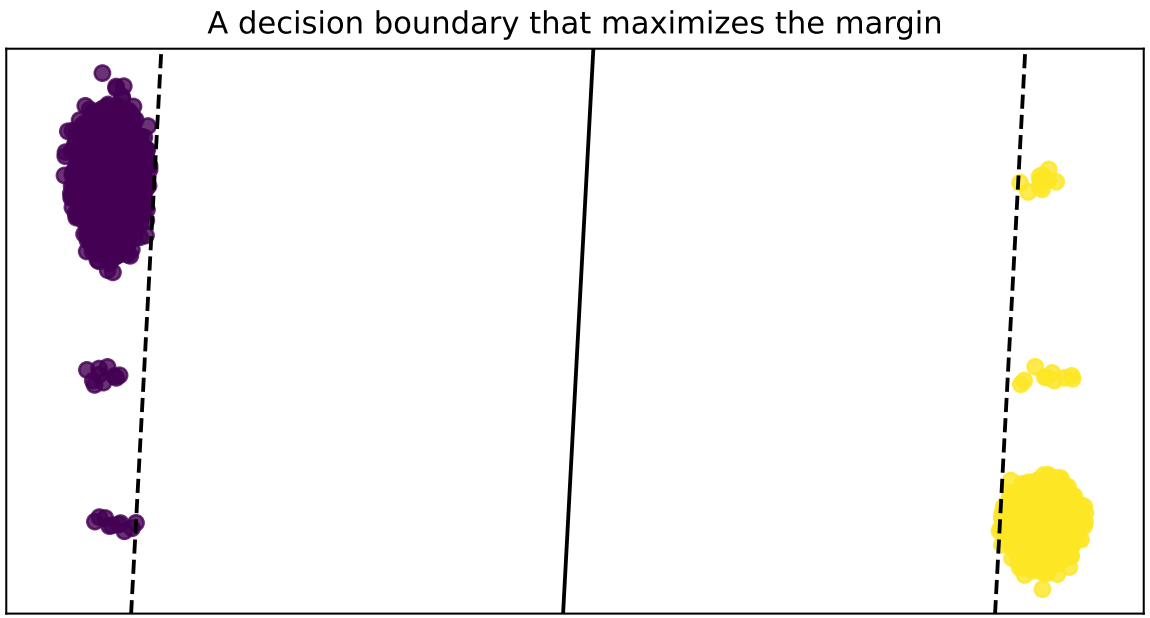
> Decision boundary may still be suboptimal.

Margin = distance from the boundary to the nearest data point



> Decision boundary may still be suboptimal.

What we might want instead...



Larger margin $\approx$ better generalization

SVM primal formulation

Lets formulate the problem as margin maximization.

For now, we will assume the following:

- linearly separable dataset $\mathcal{D} \ni (\boldsymbol{x}_i, t_i)$
- two classes, targets $t_i \in \{-1, +1\}$
- $y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$

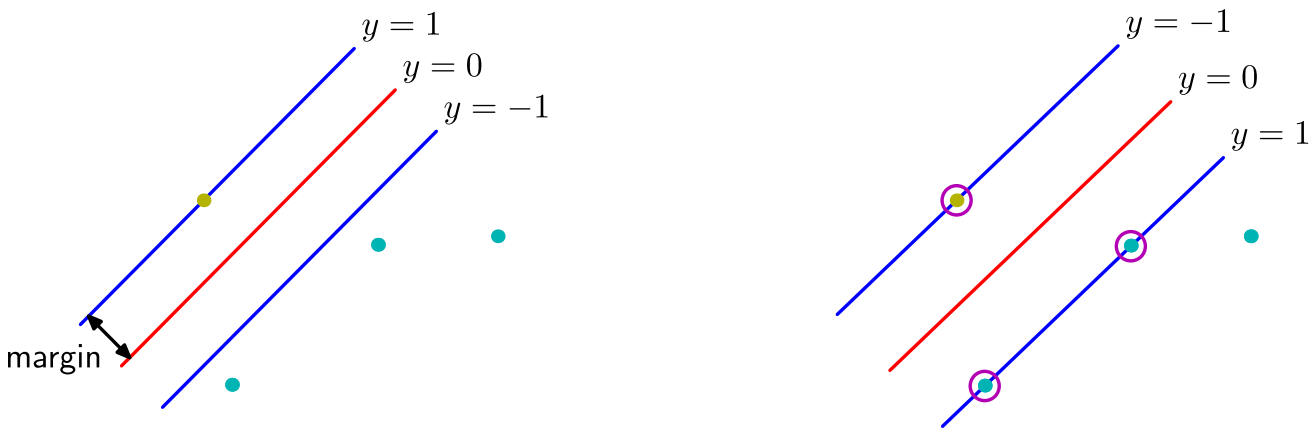


Figure 7.1 of Pattern Recognition and Machine Learning.

SVM primal formulation

Assume we have a decision boundary given by $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ that perfectly separates our training data, i.e. $t_i y(\boldsymbol{x}_i) > 0$ for all i.

The distance of a given point x_i from the decision boundary is:

$$d_i = \frac{|\boldsymbol{w}^T\boldsymbol{x}_i + b|}{\|\boldsymbol{w}\|} = \frac{t_i y(\boldsymbol{x_i})}{\|\boldsymbol{w}\|}$$

The margin m is the distance of the point closest to the decision boundary:

$$m = \min_i d_i = \frac{\min_i t_i y(\boldsymbol{x_i})}{\|\boldsymbol{w}\|}$$

SVM primal formulation

Notice if we multiply the weights and bias by a positive constant $c > 0$:

$$\boldsymbol{w}' = c \cdot \boldsymbol{w}, b' = c \cdot b$$

we do not change the decision boundary they describe. The values $y'(\boldsymbol{x}_i) = \boldsymbol{w}'^T \boldsymbol{x}_i + b'$ this have the same sign, but their magnitude changes:

$$y'(\boldsymbol{x_i}) = c \cdot y(\boldsymbol{x_i})$$

We will set c such that $t_i y(\boldsymbol{x}_i) = 1$ for the closest point x_i to the decision boundary.

In other words $\min_i t_i y(\boldsymbol{x}_i) = 1$, but also:

$$t_i y(\boldsymbol{x}_i) \geq 1$$

SVM primal formulation

With $\min_i t_i y(\boldsymbol{x}_i) = 1$, we get that the margin is:

$$m = \frac{\min_i t_i y(\boldsymbol{x_i})}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}$$

We want to find $\boldsymbol{w}, b$ that maximize the margin and satisfy $t_i y(\boldsymbol{x}_i) \geq 1 \quad \forall i$:

$$\boldsymbol{w}^*, b^* = \operatorname*{argmax}_{\boldsymbol{w},b} \frac{1}{\|\boldsymbol{w}\|} \qquad \text{s.t.:} \quad t_i y(\boldsymbol{x}_i) - 1 \geq 0 \quad \forall i$$

Or equivalently...

$$\boldsymbol{w}^*, b^* = \operatorname*{argmin}_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 \qquad \text{s.t.:} \quad t_i y(\boldsymbol{x}_i) - 1 \geq 0 \quad \forall i$$

SVM primal formulation

What we have arrived at:

$$\boldsymbol{w}^*, b^* = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2 \qquad \text{s.t.:} \quad t_i y(\boldsymbol{x}_i) - 1 \geq 0 \quad \forall i$$

This is called the primal formulation of the SVM learning problem.

Compare it to the formulation of logistic regression for which we used SGD:

$$\boldsymbol{w}^*, b^* = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{N} \sum_i \mathcal{L}(t_i, y(\boldsymbol{x}_i))$$

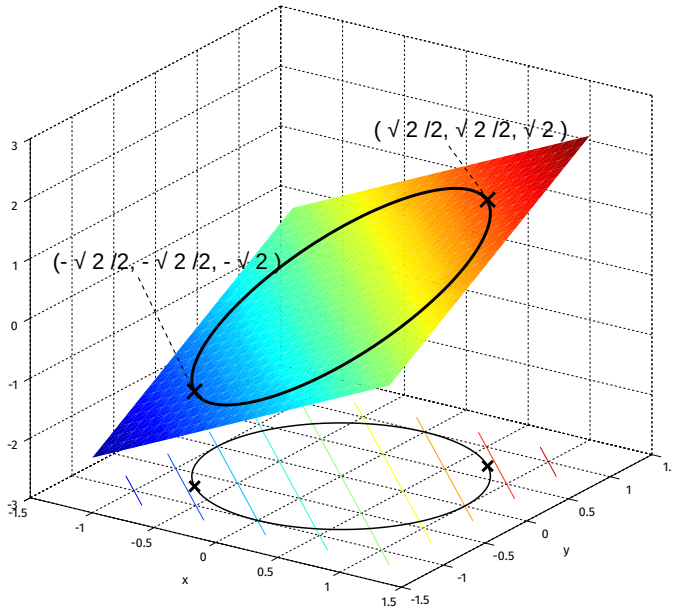
This time we cannot just use SGD though. We need another tool.

Constrained Optimization - Equality Constraints

a) Given a function $f(\boldsymbol{x})$, we want to find its minimum w.r.t. a vector $\boldsymbol{x}$.

   $\Rightarrow$ We investigate the critical points: $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$

b)



https://commons.wikimedia.org/wiki/File:Lagrange_very_simple.svg

Constrained Optimization - Equality Constraints

a) Given a function $f(\boldsymbol{x})$, we want to find its minimum w.r.t. a vector $\boldsymbol{x}$.

   $\Rightarrow$ We investigate the critical points: $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$

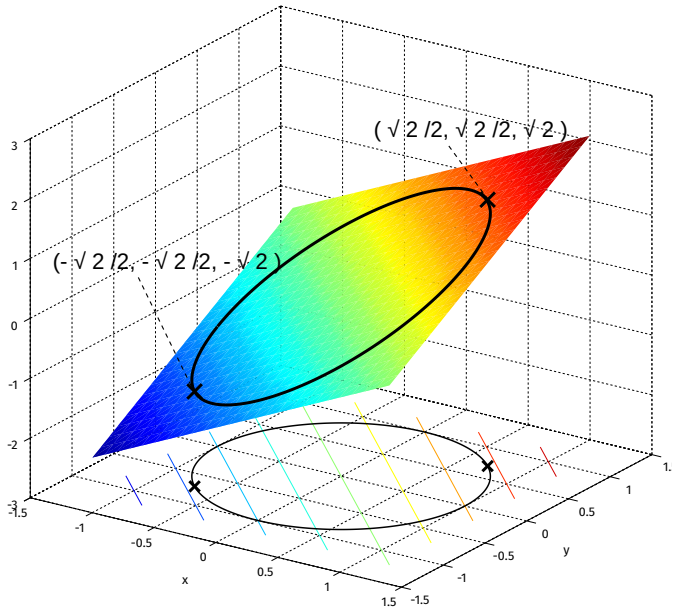
b) We can incorporate N equality constraints $g_i(\boldsymbol{x}) = 0$ that the solution must satisfy.

   $\Rightarrow$ We introduce N **Lagrange multipliers** $\boldsymbol{\lambda} \in \mathbb{R}^N$, and we form a Lagrangian:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_i \lambda_i g_i(\boldsymbol{x}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^T g(\boldsymbol{x})$$

   And again, we investigate its critical points:

$$\nabla_{\boldsymbol{x},\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = 0$$



https://commons.wikimedia.org/wiki/File:Lagrange_very_simple.svg

Constrained Optimization - Inequality Constraints

c) In our case, we have N inequality constraints $g_i(\boldsymbol{x}) \geq 0$.

We again start by forming a Lagrangian:

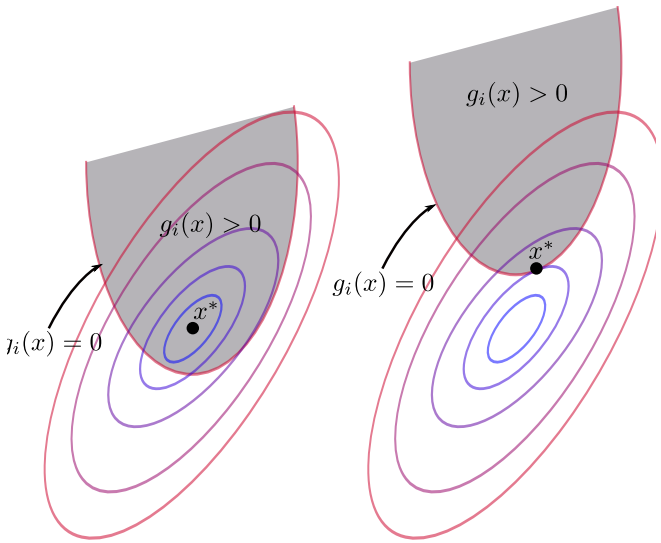
$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^T g(\boldsymbol{x})$$

This time we'll investigate only the critical points in $\boldsymbol{x}$:

$$\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = 0$$

Now, there are two cases to examine for each constraint.

Consider first a simpler case with a convex function $f(\boldsymbol{x})$, and a single constraint $g(\boldsymbol{x}) \geq 0$.



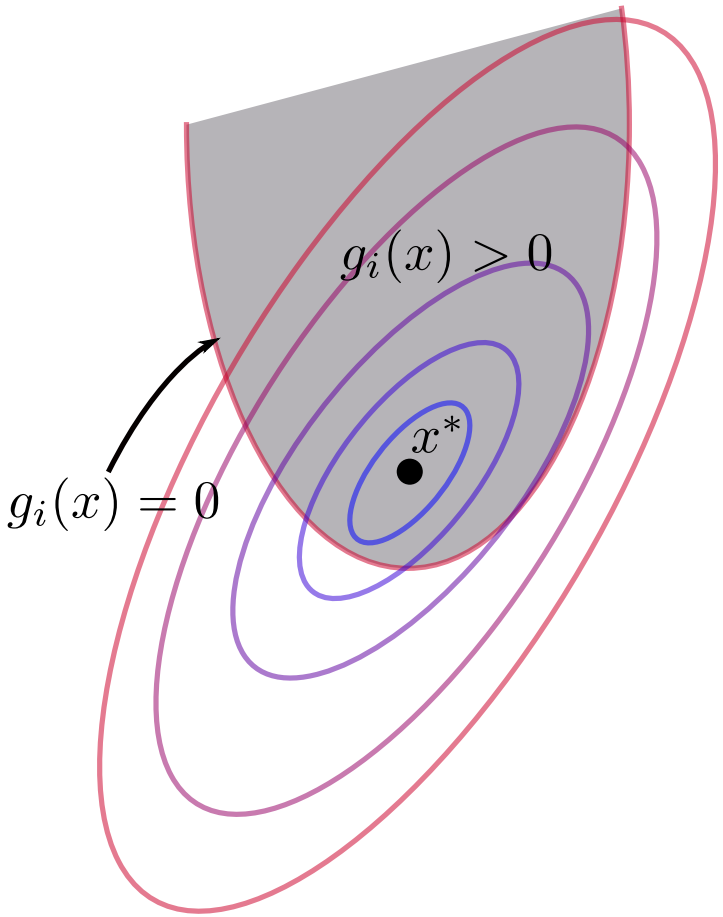
https://upload.wikimedia.org/wikipedia/commons/5/5d/Inequality_constraint_diagram.svg

Constrained Optimization - Inequality Constraints

Case 1)

If the optimum of $f(\boldsymbol{x})$ lies within the region $g(\boldsymbol{x}) > 0$, the constraint is said to be **inactive**.

We would find the optimum even if we'd remove the constrain, i.e. set $\lambda = 0$.



https://upload.wikimedia.org/wikipedia/commons/5/5d/Inequality_constraint_diagram.svg

Constrained Optimization - Inequality Constraints

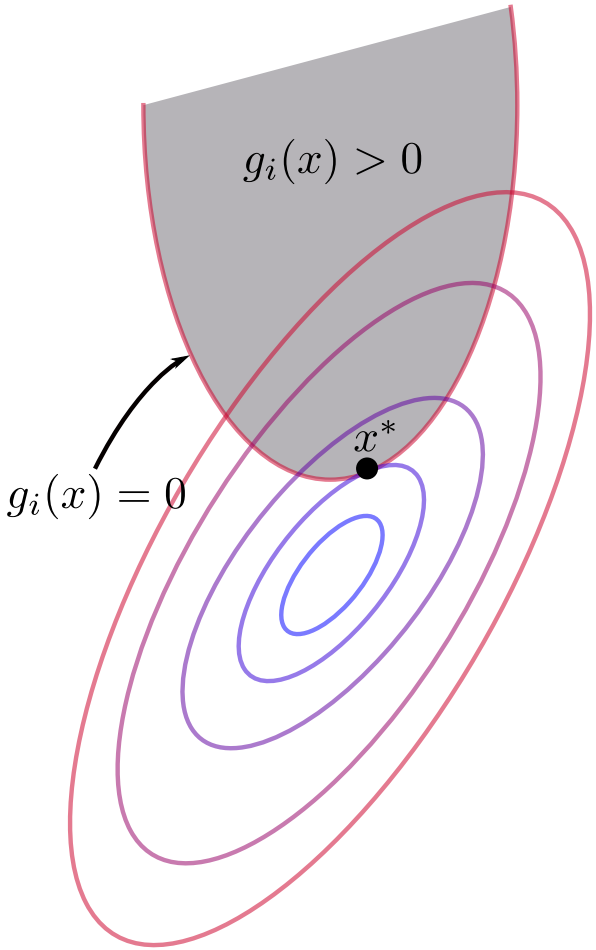
Case 2)

If the actual optimum is outside of the region, then we are looking for a minimum on the boundary $g(\boldsymbol{x}) = 0$.

The constraint is said to be **active**.

The minimum corresponds to a critical point with $\lambda \neq 0$.

Notice that in both cases, $\lambda g(\boldsymbol{x}) = 0$.



https://upload.wikimedia.org/wikipedia/commons/5/5d/Inequality_constraint_diagram.svg

Constrained Optimization - Inequality Constraints

Case 2)

The minimum corresponds to a point on the boundary.

Notice that in both cases, $\lambda g(\boldsymbol{x}) = 0$.

This time, the sign of λ matters. We can use the fact, that $\nabla g(\boldsymbol{x})$ on the boundary is oriented **into** the region $g(\boldsymbol{x}) \geq 0$. Minimum is attained only when the gradient of $f(\boldsymbol{x})$ is also oriented **into** the region. We therefore require:

$$\nabla f(\boldsymbol{x}) = \lambda \nabla g(\boldsymbol{x}) \text{ for } \lambda > 0$$

Again, notice that in both cases, $\lambda \geq 0$.

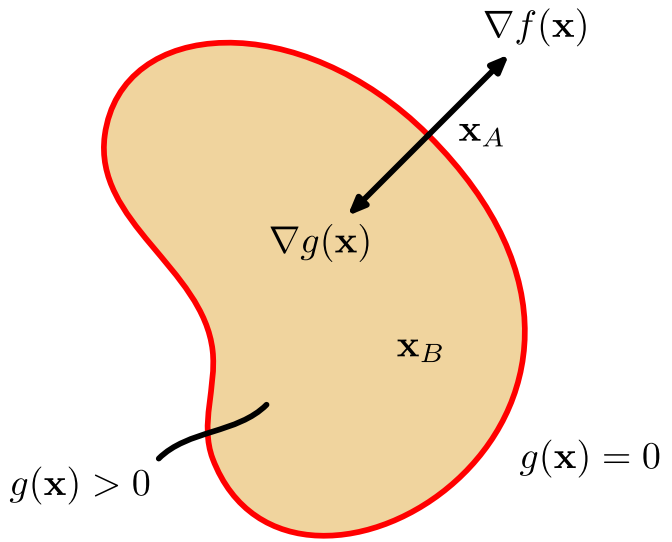


Figure E.3 of Pattern Recognition and Machine Learning.

Minimization - Inequality Constraints

Theorem: Let $f(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}$ be a function, which has a minimum in $\boldsymbol{x}^*$ subject to N inequality constraints $g_i(\boldsymbol{x}) \geq 0$. Assume that all f and g_i have continuous partial derivatives, and that $\nabla_{\boldsymbol{x}} g_i(\boldsymbol{x}^*) \neq 0$.

Then, there exist $\boldsymbol{\lambda} \in \mathbb{R}^N$, such that the **Lagrangian function**

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \stackrel{\text{def}}{=} f(\boldsymbol{x}) - \boldsymbol{\lambda}^T g(\boldsymbol{x})$$

has zero gradient in $\boldsymbol{x}^*$, and the following conditions hold $\forall i$:

$$g_i(\boldsymbol{x}) \geq 0,$$

$$\lambda_i \geq 0,$$

$$\lambda_i g_i(\boldsymbol{x}) = 0.$$

These conditions are known as **Karush-Kuhn-Tucker (KKT)** conditions.

Necessary and Sufficient KKT Conditions

The KKT conditions are **necessary** conditions for a minimum.

- i.e. the minimum has to satisfy them, but other points can satisfy them as well.

However, in the following settings, the conditions are also **sufficient**:

- we are optimizing a *convex* function $f(\boldsymbol{x})$
- the inequality constraints are continuously differentiable convex functions
- the equality constraints (if any) are affine functions

Our problem satisfies these settings:

$$\boldsymbol{w}^*, b^* = \underset{\boldsymbol{w},b}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2 \qquad \text{s.t.:} \quad t_i y(\boldsymbol{x}_i) - 1 \geq 0 \quad \forall i$$

Minimization - Inequality Constraints

Lemma (Lagrangian duality): For a problem that is in the settings from previous slide, if $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^T g(\boldsymbol{x})$ has a minimum in $\boldsymbol{x}^*$, then:

$\boldsymbol{\lambda}$ fulfills the KKT conditions $\iff \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda})$ has a maximum in $\boldsymbol{\lambda}$ subject to $\boldsymbol{\lambda} \geq 0$

Minimization - Inequality Constraints

Lemma (Lagrangian duality): For a problem that is in the settings from previous slide, if $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^T g(\boldsymbol{x})$ has a minimum in $\boldsymbol{x}^*$, then:

$\boldsymbol{\lambda}$ fulfills the KKT conditions $\iff \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda})$ has a maximum in $\boldsymbol{\lambda}$ subject to $\boldsymbol{\lambda} \geq 0$

Proof:

$\Rightarrow$) We have $\boldsymbol{\lambda}$ s.t. $g_i(\boldsymbol{x}^*) \geq 0, \lambda_i \geq 0, \lambda_i g_i(\boldsymbol{x}^*) = 0$.

- if $g_i(\boldsymbol{x}^*) = 0$, then $\mathcal{L}$ does not change with changing λ_i
- if $g_i(\boldsymbol{x}^*) > 0$, then $\lambda_i = 0$ from the KKT cond., which maximizes $\mathcal{L}$ subject to $\lambda_i \geq 0$

$\Leftarrow$)

Minimization - Inequality Constraints

Lemma (Lagrangian duality): For a problem that is in the settings from previous slide, if $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^T g(\boldsymbol{x})$ has a minimum in $\boldsymbol{x}^*$, then:

$\boldsymbol{\lambda}$ fulfills the KKT conditions $\iff \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda})$ has a maximum in $\boldsymbol{\lambda}$ subject to $\boldsymbol{\lambda} \geq 0$

Proof:

$\Rightarrow$) We have $\boldsymbol{\lambda}$ s.t. $g_i(\boldsymbol{x}^*) \geq 0, \lambda_i \geq 0, \lambda_i g_i(\boldsymbol{x}^*) = 0$.

- if $g_i(\boldsymbol{x}^*) = 0$, then $\mathcal{L}$ does not change with changing λ_i
- if $g_i(\boldsymbol{x}^*) > 0$, then $\lambda_i = 0$ from the KKT cond., which maximizes $\mathcal{L}$ subject to $\lambda_i \geq 0$

$\Leftarrow$) We have a maximum of $\mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\lambda})$ in $\boldsymbol{\lambda}$ subject to $\lambda_i \geq 0$.

- if $g_i(\boldsymbol{x}^*) < 0$, then increasing λ_i would increase $\mathcal{L}$ ⚡
- if $g_i(\boldsymbol{x}^*) > 0$, then decreasing λ_i increases $\mathcal{L}$, so $\lambda_i = 0$ □

Dual formulation derivation

Problem:

$$w^*, b^* = \operatorname*{argmin}_{w,b} \frac{1}{2}\|w\|^2$$

Lagrangian:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i (t_i y(x_i) - 1)$$

KKT conditions:

$$t_i y(x_i) - 1 \geq 0$$

$$\alpha_i \geq 0$$

$$\alpha_i (t_i y(x_i) - 1) = 0$$

Dual formulation derivation

We have the Lagrangian:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_i \alpha_i (t_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1)$$

For easier manipulation, let's rewrite it as:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\sum_j w_j^2 - \sum_i \sum_j \alpha_i t_i w_j x_{i,j} - \sum_i \alpha_i t_i b + \sum_i \alpha_i$$

Then we set these partial derivatives to 0:

$$0 = \frac{\partial}{\partial w_j}\mathcal{L} = w_j - \sum_i \alpha_i t_i x_{i,j}, \qquad 0 = \frac{\partial}{\partial b}\mathcal{L} = -\sum_i \alpha_i t_i$$

We get: $w = \sum_i \alpha_i t_i \boldsymbol{x}_i$, and $\sum_i \alpha_i t_i = 0$

Dual formulation derivation

Substituting $w = \sum_i \alpha_i t_i \boldsymbol{x}_i$ back into the Lagrangian:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_i \alpha_i (t_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1)$$

We get:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\left(\sum_i \alpha_i t_i \boldsymbol{x}_i\right)^T \left(\sum_j \alpha_j t_j \boldsymbol{x}_j\right) - \sum_i \alpha_i \left(t_i \left(\left(\sum_j \alpha_j t_j \boldsymbol{x}_j\right)^T \boldsymbol{x}_i + b\right) - 1\right)$$

This simplifies to:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j - b\sum_i \alpha_i t_i + \sum_i \alpha_i$$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j - b\sum_i \alpha_i t_i + \sum_i \alpha_i$$

We know $\sum_i \alpha_i t_i = 0$, so the b term vanishes:

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

We know from the **Theorem**, that if we find $\boldsymbol{\alpha}$ that satisfy the KKT conditions, we can express:

$$\boldsymbol{w}^* = \sum_i \alpha_i t_i x_i$$

Furthermore, $b^* = t_s - \boldsymbol{w}^{*T} x_s$ for any $\alpha_s > 0$, because $\alpha_s(t_s y(\boldsymbol{x}_s) - 1) = 0$.

We must now find α^* that satisfy the KKT conditions. From the **Lemma**, we know that such α^* maximize the Lagrangian $\mathcal{L}(\boldsymbol{\alpha})$ subject to $\alpha_i \geq 0$. Therefore, we will find them as:

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j \qquad \text{s.t. } \alpha_i \geq 0$$

But, during derivation we also found the condition $\sum_i \alpha_i t_i = 0$.

Dual formulation derivation

Problem:

$$\boldsymbol{w}^*, b^* = \operatorname*{argmin}_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$

Lagrangian:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_i \alpha_i (t_i y(\boldsymbol{x}_i) - 1)$$

KKT conditions:

$$t_i y(\boldsymbol{x}_i) - 1 \geq 0$$

$$\alpha_i \geq 0$$

$$\alpha_i (t_i y(\boldsymbol{x}_i) - 1) = 0$$

Dual problem:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

Conditions:

$$\alpha_i \geq 0$$

$$\sum_i \alpha_i t_i = 0$$

Dual formulation key points

Dual problem:

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

Inference: $y(\boldsymbol{x}) = \sum_i \alpha_i t_i \boldsymbol{x}_i^T \boldsymbol{x} + b$

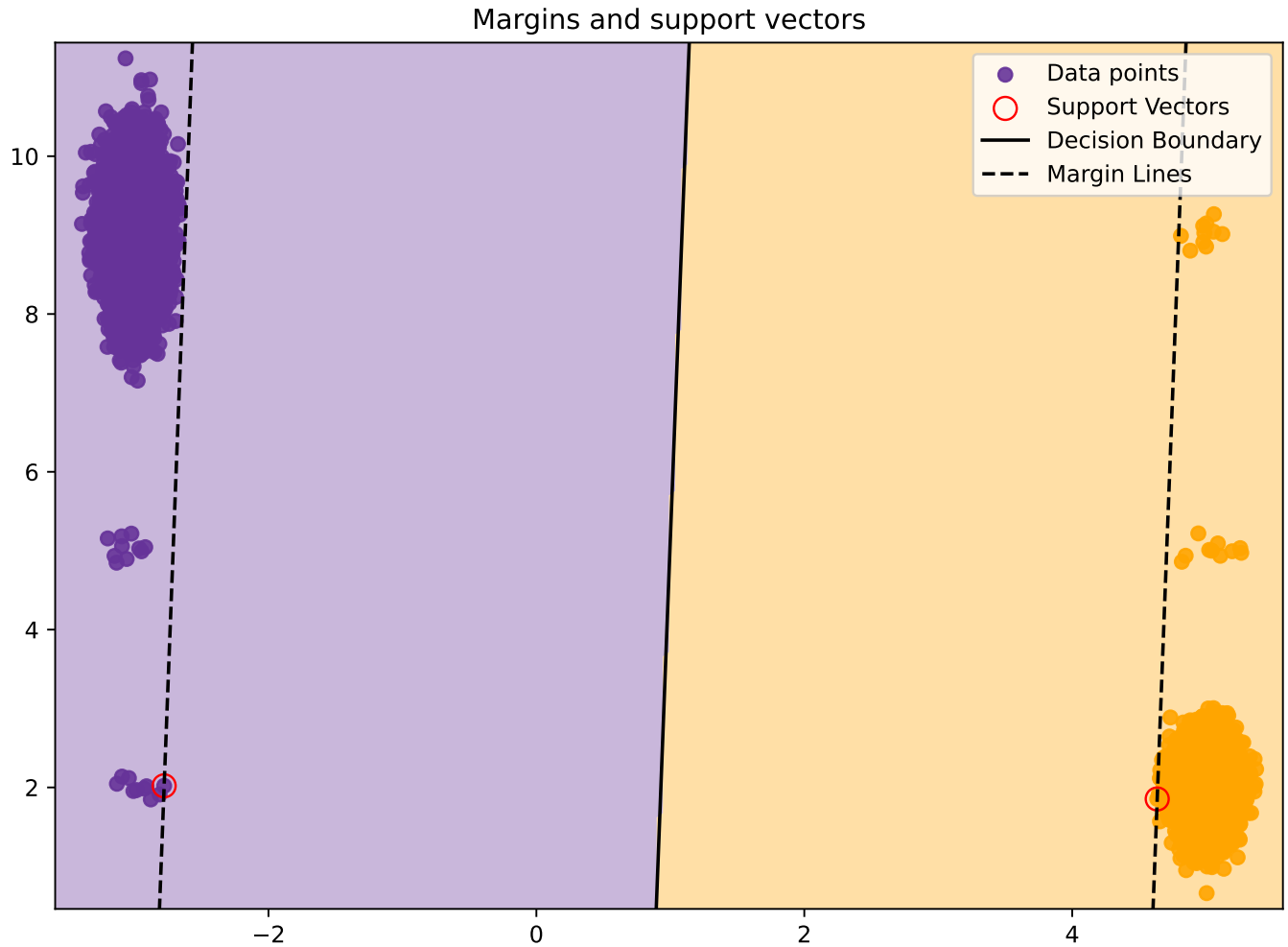
Conditions:

$$\alpha_i \geq 0$$

$$\sum_i \alpha_i t_i = 0$$

- Thanks to the lemma, we don't need to explicitly satisfy the KKTs. We know they will be satisfied for $\boldsymbol{\alpha}^*$.
- From the KKTs, we know that $\alpha_i(t_i y(\boldsymbol{x_i}) - 1) = 0$. For most samples x_i, we'll actually have $\alpha_i = 0$. Those with $\alpha_i > 0$ are the **support vectors**.

Support vectors in a linearly-separable dataset



Dual formulation key points

Dual problem:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

Inference: $y(\boldsymbol{x}) = \sum_i \alpha_i t_i \boldsymbol{x}_i^T \boldsymbol{x} + b$

Conditions:

$$\alpha_i \geq 0$$

$$\sum_i \alpha_i t_i = 0$$

- $b = t_s - \boldsymbol{w}^T \boldsymbol{x}_s$ for any support vector $\boldsymbol{x}_s$. Usually a mean over all support vectors is computed for numerical stability.

Dual formulation key points

Dual problem:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

Inference: $y(\boldsymbol{x}) = \sum_i \alpha_i t_i \boldsymbol{x}_i^T \boldsymbol{x} + b$

Conditions:

$$\alpha_i \geq 0$$

$$\sum_i \alpha_i t_i = 0$$

- $b = t_s - \boldsymbol{w}^T \boldsymbol{x}_s$ for any support vector $\boldsymbol{x}_s$. Usually a mean over all support vectors is computed for numerical stability.
- Instead of $\boldsymbol{w}$, we use the nonzero α_i and their support vectors for inference.

Dual formulation key points

Dual problem:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

Inference: $y(\boldsymbol{x}) = \sum_i \alpha_i t_i \boldsymbol{x}_i^T \boldsymbol{x} + b$

Conditions:

$$\alpha_i \geq 0$$

$$\sum_i \alpha_i t_i = 0$$

- $b = t_s - \boldsymbol{w}^T \boldsymbol{x}_s$ for any support vector $\boldsymbol{x}_s$. Usually a mean over all support vectors is computed for numerical stability.
- Instead of $\boldsymbol{w}$, we use the nonzero α_i and their support vectors for inference.
- The feature vectors $\boldsymbol{x}$ are only used in the scalar products $\boldsymbol{x}_i^T \boldsymbol{x}_j$. We can replace them with any **kernel function** $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Kernels

Definition A symmetric function $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is called a positive-definite kernel, if:

$\forall \boldsymbol{x}_1, ..., \boldsymbol{x}_n \in \mathbb{R}^D, \forall c_1, ..., c_n \in \mathbb{R}$:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

- This definition holds for the skalar product of feature vectors with any mapping φ:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \varphi(\boldsymbol{x})^T \varphi(\boldsymbol{z})$$

Other commonly used kernels

- **Polynomial kernel of degree d**

$$K(\boldsymbol{x}, \boldsymbol{z}) = \left(\gamma \boldsymbol{x}^T \boldsymbol{z}\right)^d$$

Other commonly used kernels

- **Polynomial kernel of degree d**

$$K(\boldsymbol{x}, \boldsymbol{z}) = \left(\gamma \boldsymbol{x}^T \boldsymbol{z}\right)^d$$

- **Polynomial kernel of degree at most d**

$$K(\boldsymbol{x}, \boldsymbol{z}) = \left(\gamma \boldsymbol{x}^T \boldsymbol{z} + 1\right)^d$$

Other commonly used kernels

- **Polynomial kernel of degree d**

$$K(\boldsymbol{x}, \boldsymbol{z}) = \left(\gamma \boldsymbol{x}^T \boldsymbol{z}\right)^d$$

- **Polynomial kernel of degree at most d**

$$K(\boldsymbol{x}, \boldsymbol{z}) = \left(\gamma \boldsymbol{x}^T \boldsymbol{z} + 1\right)^d$$

- **Gaussian Radial basis function (RBF)**

$$K(\boldsymbol{x}, \boldsymbol{z}) = e^{-\gamma \left\|\boldsymbol{x}-\boldsymbol{z}\right\|^2}$$

Dual problem key points

Dual problem:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Inference: $y(\boldsymbol{x}) = \sum_i \alpha_i t_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b$

Conditions:

$$\alpha_i \geq 0$$

$$\sum_i \alpha_i t_i = 0$$

- The kernels are cheaper to compute, than the actual polynomial mappings.

Dual problem key points

Dual problem:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Inference: $y(\boldsymbol{x}) = \sum_i \alpha_i t_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b$

Conditions:

$$\alpha_i \geq 0$$

$$\sum_i \alpha_i t_i = 0$$

- The kernels are cheaper to compute, than the actual polynomial mappings.
- The RBF kernel is a function of distance. SVM with the RBF kernel can be viewed as an extended version of the k-nearest neighbors algorithm, selecting only the most significant samples and weights them by similarity.

Dual problem key points

Dual problem:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Inference: $y(\boldsymbol{x}) = \sum_i \alpha_i t_i K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b$

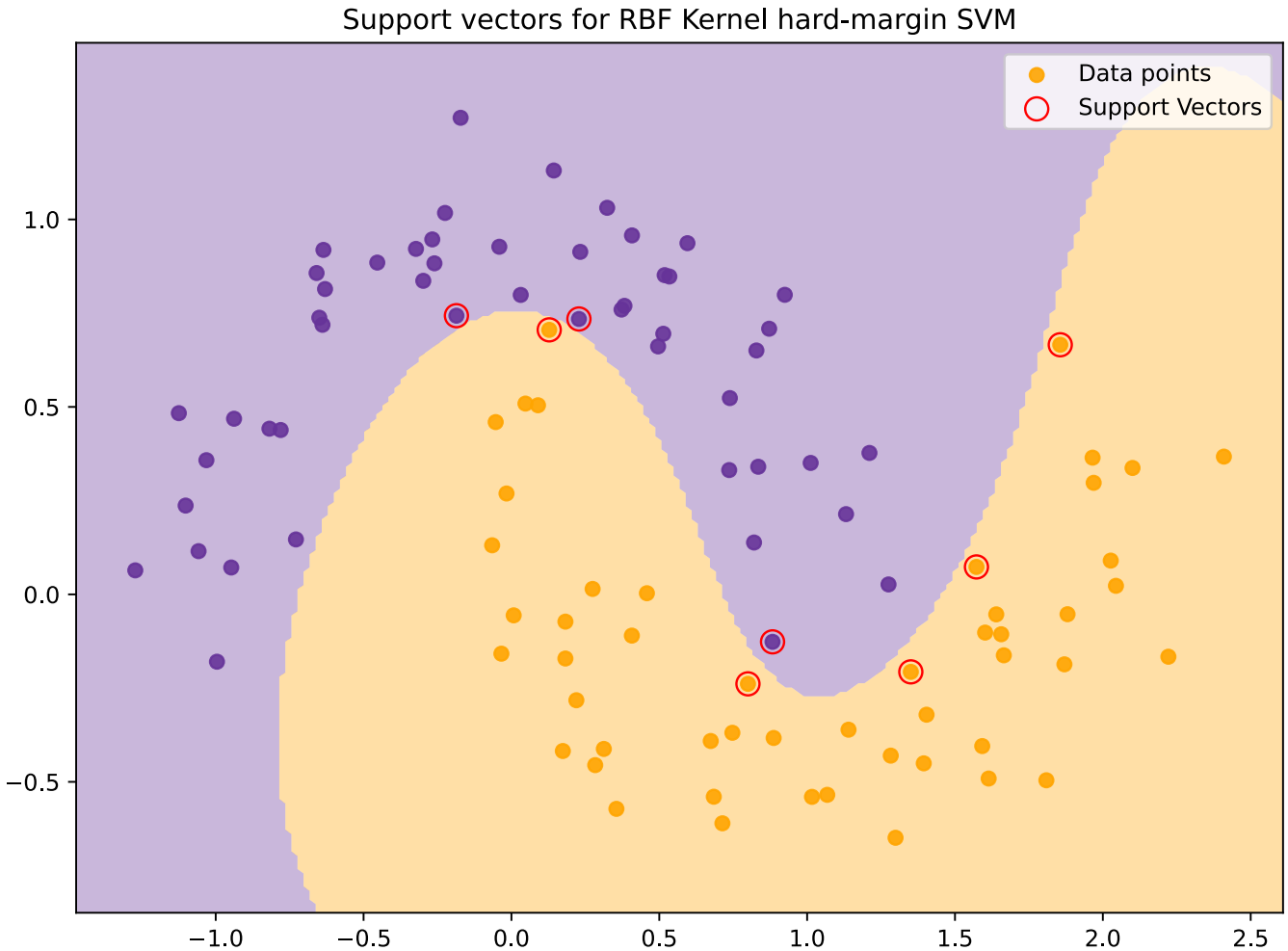
Conditions:

$$\alpha_i \geq 0$$

$$\sum_i \alpha_i t_i = 0$$

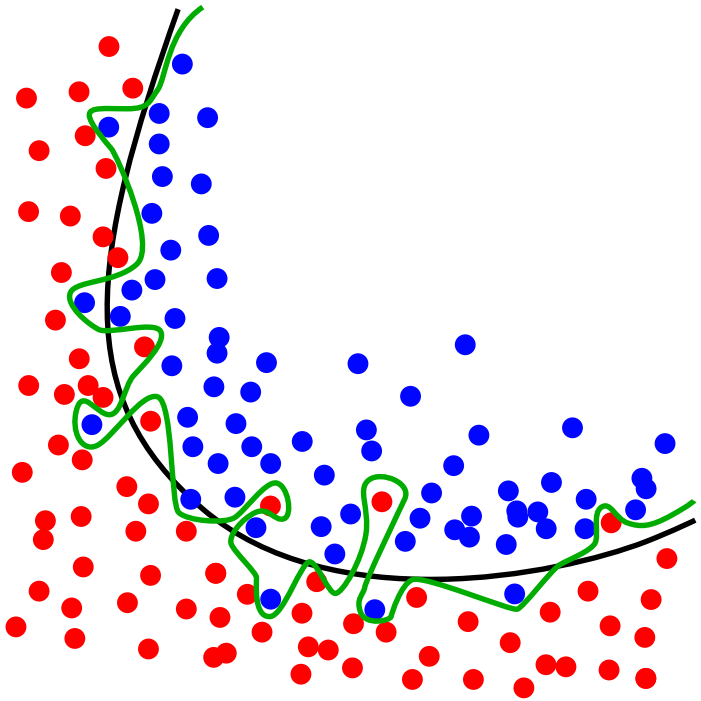
- The kernels are cheaper to compute, than the actual polynomial mappings.
- The RBF kernel is a function of distance. SVM with the RBF kernel can be viewed as an extended version of the k-nearest neighbors algorithm, selecting only the most significant samples and weights them by similarity.
- The α_i multipliers are given by the data. SVMs are considered to be non-parametric.

RBF Kernel support vectors in a non-linear dataset



Overfitting

SVMs with RBF kernel can technically separate any dataset. This typically causes overfitting on noise.



https://upload.wikimedia.org/wikipedia/commons/1/19/Overfitting.svg

Allowing SVMs to make mistakes ⇒ Soft-margin SVMs

Until now, we assumed the data do be linearly separable, and allowed points to only be outside the margin.

We will now relax this condition, and allow points to be **in the margin**, or even on the **wrong side** of the decision boundary.

We introduce **slack variables** $\xi_i \geq 0$ for each training instance i.

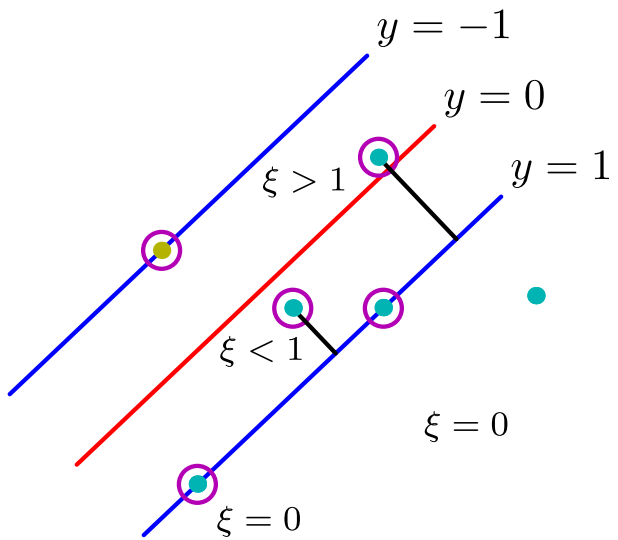


Figure 7.3 of Pettern Recognition and Machine Learning

We will define them such that $\xi_i = 0$ signifies a point outside of the margin, $\xi_i \in (0, 1)$ for a point inside the margin, $\xi_i = 1$ for a point on the decision boundary, and $\xi_i > 1$ for the wrong side of the boundary. We want $\sum_i \xi_i$ to be minimal.

Soft-margin SVM derivation

Problem:

$$\boldsymbol{w}^*, b^* = \operatorname*{argmin}_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$

Lagrangian:

$$\mathcal{L} = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i(t_i y(\boldsymbol{x}_i) + \xi_i - 1) - \sum_i \beta_i \xi_i$$

KKT conditions:

$$t_i y(\boldsymbol{x}_i) + \xi_i - 1 \geq 0, \xi_i \geq 0$$

$$\alpha_i(t_i y(\boldsymbol{x}_i) + \xi_i - 1) = 0, \alpha_i \geq 0$$

$$\beta_i \xi_i = 0, \beta_i \geq 0$$

Soft-margin SVM derivation

We perform the same procedure as before with the Lagrangian:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i(t_i y(\boldsymbol{x}_i) + \xi_i - 1) - \sum_i \beta_i \xi_i$$

This time we also compute the partial derivative w.r.t. ξ_i:

$$\frac{\partial}{\partial \xi_i}\mathcal{L} = C - \alpha_i - \beta_i = 0$$

Again, substituting everything back into the Lagrangian we obtain:

$$\mathcal{L} = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j + \sum_i \xi_i (C - \alpha_i - \beta_i)$$

Since $C - \alpha_i - \beta_i = 0$, the last sum is also equal to 0.

Soft-margin SVM derivation

We have arrived to the exact same expression as before, which we now must maximize:

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j t_i t_j \boldsymbol{x_i}^T \boldsymbol{x}_j \qquad \text{s.t. } \alpha_i \geq 0, \quad \sum_i \alpha_i t_i = 0$$

But this time, we also have $C - \alpha_i - \beta_i = 0$, i.e.:

$$\alpha_i = C - \beta_i$$

Since $\beta_i \geq 0$ from the KKT, we get:

$$\alpha_i \leq C$$

Our actual constraints are therefore:

$$C \geq \alpha_i \geq 0, \quad \sum_i \alpha_i t_i = 0$$

Soft-margin SVM derivation

Problem:

$$\boldsymbol{w}^*, b^* = \underset{\boldsymbol{w},b,\boldsymbol{\xi}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$

Lagrangian:

$$\mathcal{L} = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_i \xi_i - \sum_i \alpha_i(t_i y(\boldsymbol{x}_i) + \xi_i - 1) - \sum_i \beta_i \xi_i$$

KKT conditions:

$$t_i y(\boldsymbol{x}_i) + \xi_i - 1 \geq 0, \xi_i \geq 0$$

$$\alpha_i(t_i y(\boldsymbol{x}_i) + \xi_i - 1) = 0, \alpha_i \geq 0$$

$$\beta_i \xi_i = 0, \beta_i \geq 0$$

Dual problem:

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j t_i t_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

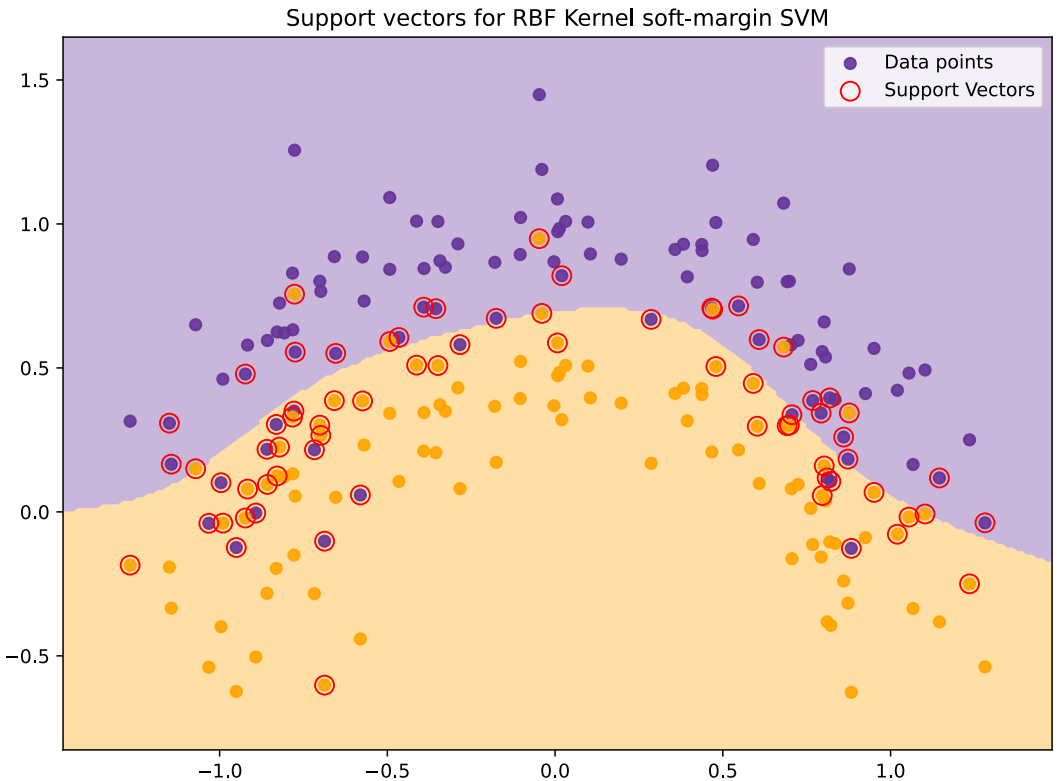
Conditions:

$$C \geq \alpha_i \geq 0$$

$$\sum_i \alpha_i t_i = 0$$

Soft-margin SVM with RBF kernel

```
svm_classifier = sklearn.svm.SVC(kernel='rbf', C=10)
```



SGD-like Formulation of Soft-Margin SVM

Note that the slack variables can be written as:

$$\xi_i = \max(0, 1 - t_i y(\boldsymbol{x}_i))$$

So, we can reformulate the soft-margin objective with the **hinge loss**:

$$\mathcal{L}_{\text{hinge}}(t, y) = \max(0, 1 - ty)$$

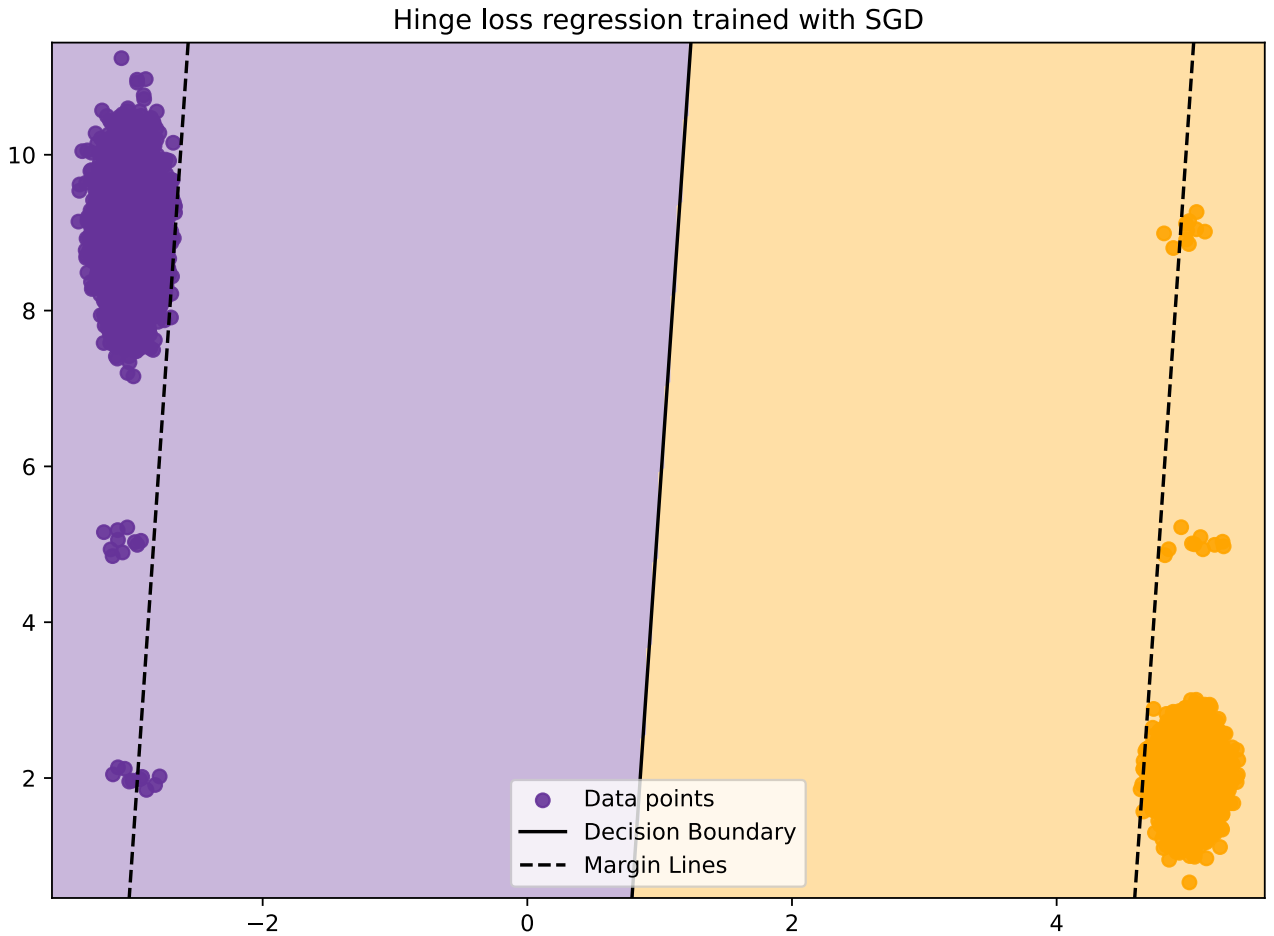
to:

$$\boldsymbol{w}^*, b^* = \operatorname*{argmin}_{\boldsymbol{w}, b} C \sum_i \mathcal{L}_{\text{hinge}}(t_i, y(\boldsymbol{x}_i)) + \frac{1}{2}\|\boldsymbol{w}\|^2$$

Which is analogous to a regularized loss, similar to logistic regression with L_2 norm:

$$\boldsymbol{w}^*, b^* = \operatorname*{argmin}_{\boldsymbol{w}, b} \frac{1}{N} \sum_i \mathcal{L}_{\text{NLL}}(t_i, y(\boldsymbol{x}_i)) + \frac{1}{2}\|\boldsymbol{w}\|^2$$

Hinge-loss regression trained by SGD



Primal versus Dual Formulation

Assume we have a dataset with N training examples, each with D features. We use a feature map φ to generate F features.

	Primal formulation	Dual formulation
Parameters	F	N
Model size	F	$s \cdot F$ for s support vectors
Usual training time	$\Theta(e \cdot N \cdot F)$ for e epochs	between $\Omega(ND)$ and $\mathcal{O}(N^2 D)$
Inference time	$\Theta(F)$	$\Theta(s \cdot D)$

Today's Lecture Objectives

After this lecture, you should be able to:

- explain the soft-margin SVM dual derivation
- know that something called “Karush-Kuhn-Tucker” conditions exists for constraint optimization
- be familiar with “kernels” and their advantages over feature maps