A.Z
Machine Learning
Nov, 2023

# lec 6 Support vector machines (SVM)

**Abstract**

This part is about SVM, soft-margin SVM, Dual-soft-margin SVM, and some optimization methods for SVM, also an introduction for kernel trick

Support vector machines (SVM)

## 1 Recall: Perceptron algorithm on linearly separable data

The upper bound $\left(\frac{2R}{\gamma}\right)^2$ of iterations of perceptron algorithm on linearly separable data

$\gamma$ : The largest achievable geometric margin in the training set, $\frac{y_i \mathbf{w}^T \mathbf{x}_i}{\|\mathbf{w}\|} \geq \gamma$ for all $i = 1 \ldots, m$

$R = \max_i \|\mathbf{x}_i\|$ : The smallest radius of the $d$-dimensional ball that encloses the training data

The perceptron algorithm is guaranteed find a consistent hyperplane if one exist. All traing data are on the correct side of the hyperplane.

However, typically there are several hyperplanes that are consistent, which one is the best?

One good solution is to choose the hyperplane $\mathbf{w}^T \mathbf{x} = 0$ that lies furthest away from the training data (maximizing the minimum geometric margin of the training examples):

$$\begin{aligned} &\text{Maximize } \gamma \\ &\text{w.r.t. variables } \mathbf{w} \in \mathbb{R}^d \\ &\text{Subject to } \frac{y_i \mathbf{w}^T \mathbf{x}_i}{\|\mathbf{w}\|} \geq \gamma, \text{for all } i = 1, \ldots, m \end{aligned}$$

SVM are based on this principle

Functional Margin: the distance from a point to the hyperplane $y_i \mathbf{w}^T \mathbf{x}_i$

Geometric Margin: the normalized version of the functional margin $\frac{y_i \mathbf{w}^T \mathbf{x}_i}{\|\mathbf{w}\|}$

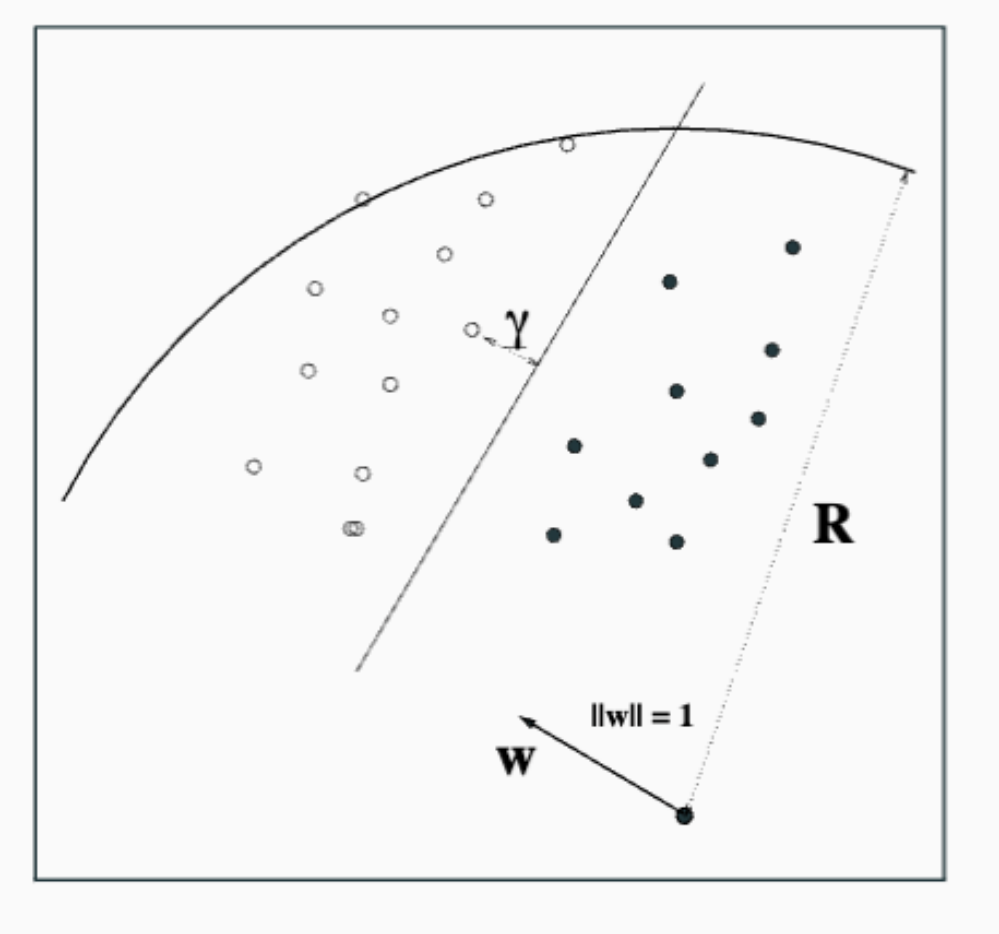


Figure 1: Example for Perceptron

in which, $\frac{y_i \mathbf{w}^T \mathbf{x}_i}{\|\mathbf{w}\|} \geq \gamma$ means for all points $x_i$, the geometric margin should be larger than $\gamma$

Multiply the constraint on the geometric margin by $\|\mathbf{w}\|$ to obtain a an equivalent constraint on the functional margin

$$y_i \mathbf{w}^T \mathbf{x}_i \geq \gamma \|\mathbf{w}\|$$

Fix the functional margin to 1: $\gamma\|\mathbf{w}\| = 1$ which gives $\gamma = \frac{1}{\|\mathbf{w}\|}$

So to maximize $\gamma$, we should minimize $\|\mathbf{w}\|$ with the constraint of having functional margin of at least 1

# 2 SVM

Hard margin support-vector machine (SVM) solves the margin maximization as follows:

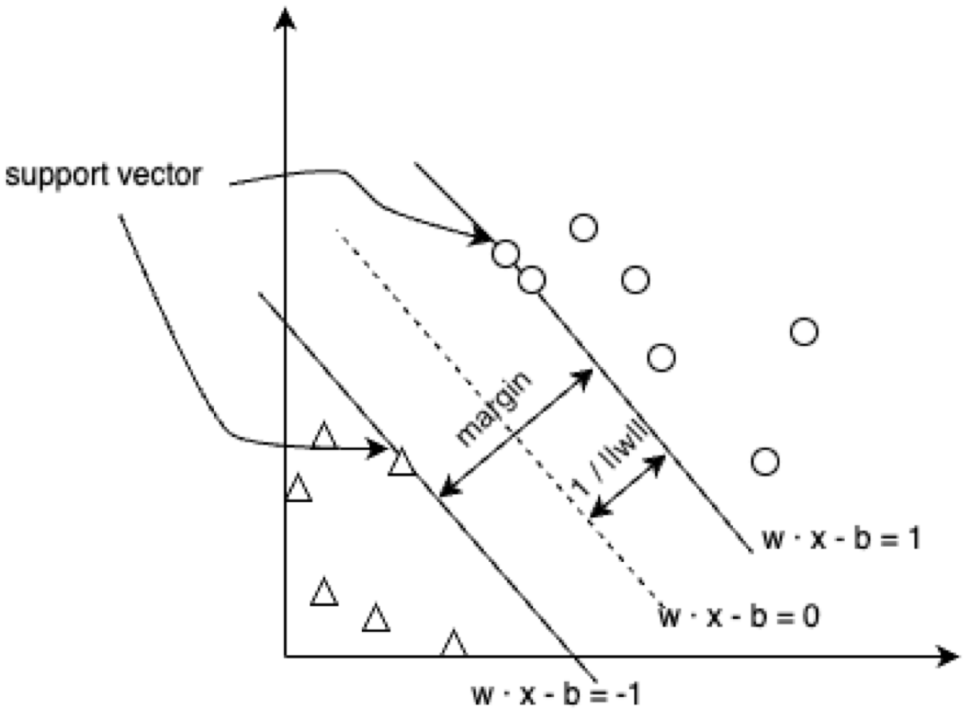


Figure 2: how select best hyperplanes?

$$
\begin{aligned}
&Minimize \quad \frac{1}{2}\|\mathbf{w}\|^2 \\
&w.r.t.\ variables \quad \mathbf{w} \in \mathbb{R}^d \\
&Subject\ to \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1, \\
&for\ all \quad i = 1, \ldots, m
\end{aligned}
$$

The points that have exactly margin $y\mathbf{w}^T\mathbf{x} = 1$ are called the support vectors

theoretical backup for maximum margin hyperplane

Consider the hypothesis class

$$\mathcal{H} = \left\{ h(\mathbf{x}) = \operatorname{sgn}\left(\mathbf{w}^T\mathbf{x}\right) \mid \min_{i=1}^{m} y_i \mathbf{w}^T \mathbf{x}_i = 1, \|\mathbf{w}\| \leq B, \|\mathbf{x}_i\| \leq R \right\}$$

which means a set of calssifiers $h(\mathbf{x}) = \operatorname{sgn}\left(\mathbf{w}^T\mathbf{x}\right)$ that satisfy certain constrains:

- Every training example $\mathbf{x}_i$ with label $y_i$ is correctly classified with a functional margin of at least 1 : $\min_{i=1}^{m} y_i \mathbf{w}^T \mathbf{x}_i = 1$.

- The norm of the weight vector $\mathbf{w}$ is bounded by $B(\|\mathbf{w}\| \leq B)$.

- The norm of each training example $\mathbf{x}_i$ is bounded by $R\,(\|\mathbf{x}_i\| \leq R)$.

The VC dimension satisfies $\text{VCdim}(\mathcal{H}) \leq B^2R^2$

Rademacher complexity satisfies: $\mathcal{R}(H) \leq \frac{RB}{\sqrt{m}}$

Thus a small norm ($\leq B$) translates to low complexity of the hypothesis class, so a better generalization error.

# 3 Soft-Margin SVM

Not all data are separable, we need to extend out model to allow misclassified training points

To allow non-separable data, we allow the functional margin of some data points to be smaller than 1 by a slack variable $\xi_i \geq 0$

The relaxed margin constraint will be expressed as

$$y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i, \xi_i \geq 0$$

$\xi_i = 0$ corresponds to having large enough margin $> 1$ $\xi_i > 1$ corresponds to negative margin, misclassified point

The set of support vectors includes all $\mathbf{x}_i$ that have non-zero slack $\xi_i$ (functional margin $\leq 1$ )

The soft-margin SVM allows non-separable data by using the relaxed the margin constraint

$$\begin{aligned}
&\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m} \xi_i \\
&\text{w.r.t variables } \mathbf{w}, \boldsymbol{\xi} \\
&\text{Subject to} \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i \\
&\text{for all} \quad i = 1, \ldots, m. \quad \xi_i \geq 0.
\end{aligned}$$

The sum (or average) of slack variables $\frac{C}{m}\sum_{i=1}^{m} \xi_i$ measure the degree of misclassification of the date points, appear as a penalty in the objective

This constraint $y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i$ indicates that each data point should be on the correct side of the margin or, if not, the degree of violation is measured b

The coefficient C $> 0$ controls the balance between model complexity (low C) and empirical error (high C).

When C is small, less emphasis is placed on the slack variables. This means that the model can tolerate more margin violations but focuses on maximizing the margin, potentially leading to a simpler model that might underfit.

When C is large, more emphasis is placed on the slack variables. This means the model

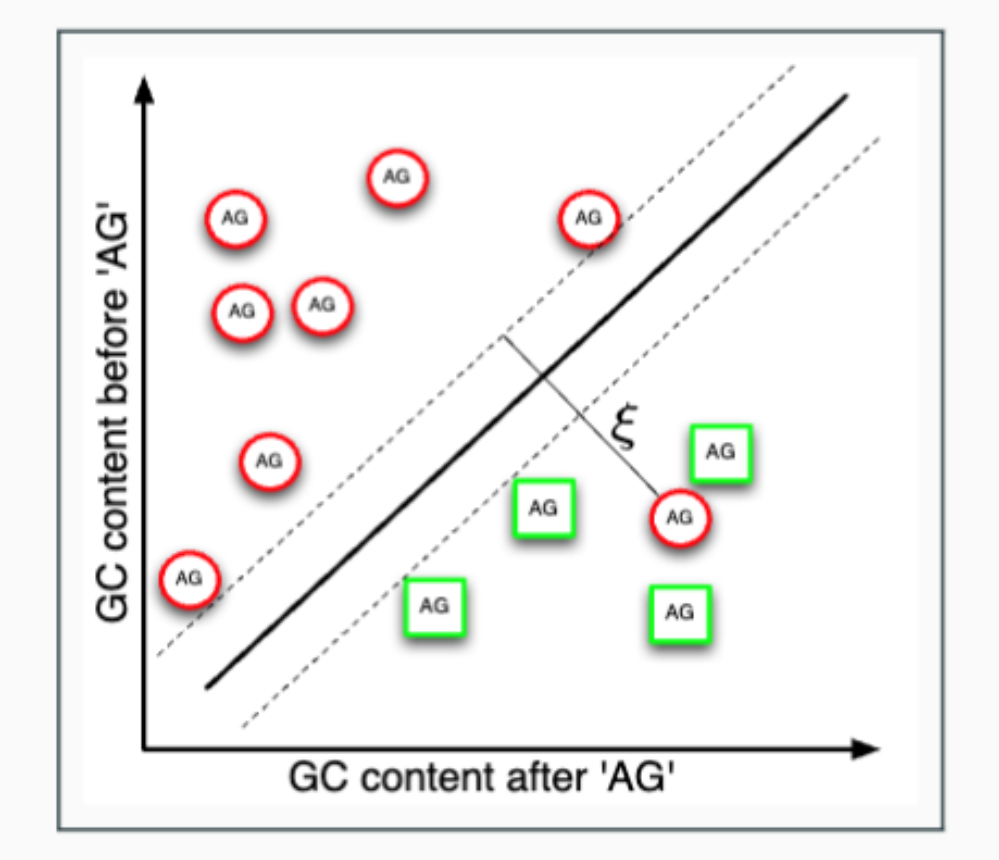


Figure 3: non-separable case

strives to reduce the number of misclassifications, even at the cost of a smaller margin, potentially leading to a more complex model that might overfit.

## 3.1 Loss function in soft-margin SVM

Observe the relaxed margin constraint:

$$y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i, \xi_i \geq 0$$

By rearranging, the same can be expressed as

$$\xi_i \geq 1 - y_i \mathbf{w}^T \mathbf{x}_i, \xi_i \geq 0$$

and further

$$\xi_i \geq \max\left(1 - y_i \mathbf{w}^T \mathbf{x}_i, 0\right)$$

The right-hand side is so called Hinge loss:

$$L_{\text{Hinge}}\ \left(y, \mathbf{w}^T\mathbf{x}\right) = \max\left(1 - y\mathbf{w}^T\mathbf{x}, 0\right)$$

and Hinge loss can be written as

$$L_{\text{Hinge}}\ (y, f(\mathbf{x})) = \max(1 - yf(\mathbf{x}), 0)$$

Hinge loss is a convex upper bound of zero-one loss

Hinge loss is zero if margin $y_i f(\mathbf{x}) \geq 1$

For a misclassified example, margin is negative and Hinge loss is

$$L_{\text{Hinge}}\ (f(\mathbf{x}), y_i)) > 1$$

The loss grows linearly in the margin violation $1 - yf(\mathbf{x})$, for margins $< 1$

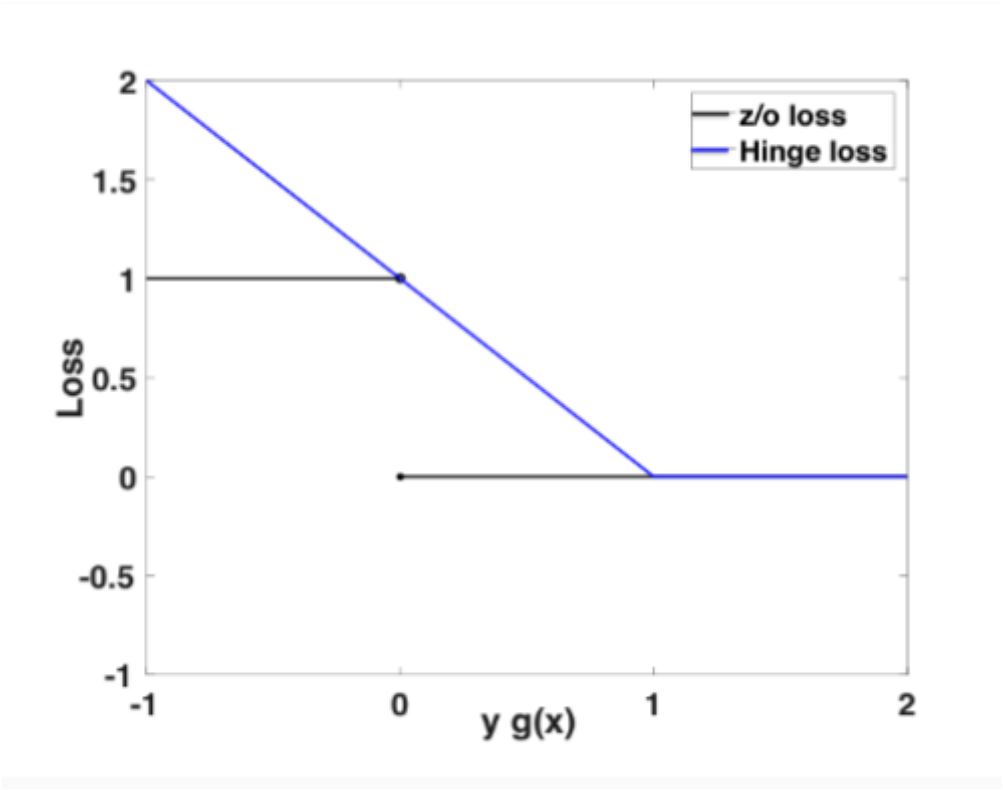


Figure 4: 0/1 loss and Hinge loss

**Interclude: differert kinds of loss function**

Because of convexity, all of the functions (except zero-loss) support fast optimizationthrough gradient based appraoches

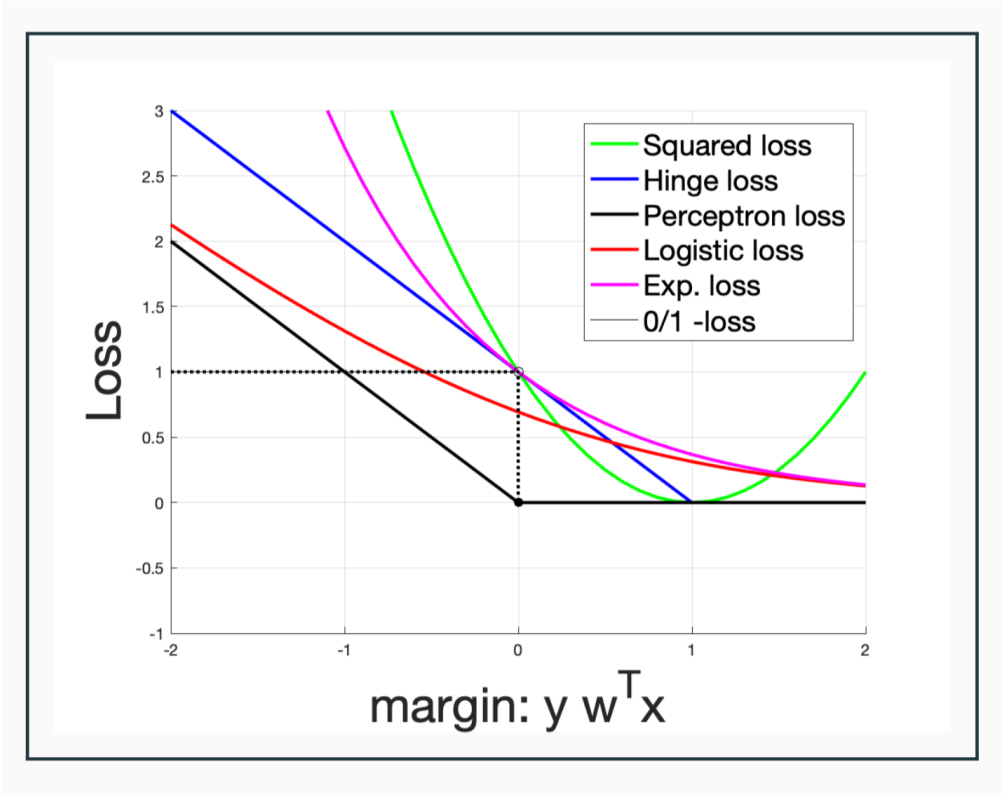


Figure 5: differert kinds of loss function

## 3.2 Optimization

Quadratic program (QP):

The objective function in a Quadratic Program forms like $\frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}$

The constraints in a Quadratic Program are linear : $A\mathbf{x} \leq \mathbf{b}$

If the matrix $Q$ is positive semi-definite, the quadratic program is convex, making it easier to solve as convex problems have a global optimum.

If $Q$ is not positive semi-definite, the problem can be non-convex and potentially NP-hard, meaning it can be computationally challenging to find the global optimum.

The soft-margin SVM corresponds to a Quadratic program

Recall soft-margin SVM problem:

$$\text{Minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{c}{m}\sum_{i=1}^{m}\xi_i$$
$$s.t. \xi_i \geq \max\left(1 - y_i\mathbf{w}^T\mathbf{x}_i, 0\right)$$
$$\text{for all i} \quad \xi_i \geq 0$$

Equvalently in terms of Hinge loss as

$$\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_{\text{Hinge}} \left( \mathbf{w}^T \mathbf{x}_i, y_i \right) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

Is so called regularized learning problem, it has two terms

- Loss Function term: Hinge loss for a data point $(\mathbf{x}_i, y_i)$ is given by $\max\left(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\right)$. The sum of the Hinge loss is a measure of the total misclassification error.

- Regularizer term: $\left(\frac{\lambda}{2}\|\mathbf{w}\|^2\right)$ , this term penalizes large weights to prevent overfitting and encourage simpler models

The parameter $\lambda$ controls the balance between the two terms

### 3.2.1 SGD for soft-margin SVM

Rewrite

$$J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} J_i(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \left( \mathcal{L}_{\text{Hinge}} \left( \mathbf{w}^T \mathbf{x}_i, y_i \right) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \right)$$

However, Hinge loss is not differentiable at point 1, so cannotsimply compute the gradient!

So we express $J_i(\mathbf{w})$ as a piecewise differentiable function

$$J_i(\mathbf{w}) = L_{\text{Hinge}} \left( \mathbf{w}^T \mathbf{x}_i, y_i \right) + \frac{\lambda}{2} \|\mathbf{w}\|^2 = \begin{cases} 1 - y_i \mathbf{w}^T \mathbf{x}_i + \frac{\lambda}{2} \|\mathbf{w}\|^2, & \text{if } y_i \mathbf{w}^T \mathbf{x}_i < 1 \\ 0 + \frac{\lambda}{2} \|\mathbf{w}\|^2 & \text{if } y_i \mathbf{w}^T \mathbf{x}_i \geq 1 \end{cases}$$

$$\nabla J_i(\mathbf{w}) = \begin{cases} -y_i \mathbf{x}_i + \lambda \mathbf{w} & \text{if } y_i \mathbf{w}^T \mathbf{x}_i < 1 \\ \mathbf{0} + \lambda \mathbf{w} & \text{if } y_i \mathbf{w}^T \mathbf{x}_i \geq 1 \end{cases}$$

Update $\mathbf{w} = \mathbf{w} - \eta \nabla J_i(\mathbf{w})$, where suggest $\eta = 1/\lambda t$

$$\mathbf{w} = \mathbf{w} - \eta \left( \lambda \mathbf{w} + \begin{cases} -y_i \mathbf{x}_i & \text{if } y_i \mathbf{w}^T \mathbf{x}_i < 1 \\ \mathbf{0} & \text{otherwise} \end{cases} \right)$$

In the case $y_i \mathbf{w}^T \mathbf{x}_i < 1$, the Hinge loss is positive, meaning the data point $(x_i, y_i)$ is misclassified or within the margin. The gradient $\nabla J_i(\mathbf{w})$ is $-y_i \mathbf{x}_i + \lambda \mathbf{w}$. The term $-y_i \mathbf{x}_i$ pushes the $\mathbf{w}$ towards correctly classifying (contrary to the current situation )

In the case $y_i \mathbf{w}^T \mathbf{x}_i \geq 1$ , the Hinge loss is zero as the data point is correctly classified. we just need to shrink $\mathbf{w}$ to make the margin larger

Compare the SVM and perceptron updata

$$\mathbf{w} = \mathbf{w} - \eta \left( \lambda \mathbf{w} + \begin{cases} -y_i \mathbf{x}_i & \text{if } y_i \mathbf{w}^T \mathbf{x}_i < 1 \\ \mathbf{0} & \text{otherwise} \end{cases} \right)$$

$$\mathbf{w} = \mathbf{w} + \begin{cases} y_i \mathbf{x}_i & \text{if } y_i \mathbf{w}^T \mathbf{x}_i < 0 \\ \mathbf{0} & \text{otherwise} \end{cases}$$

$y_i \mathbf{w}^T \mathbf{x}_i < 1$ in SVM means the data point is on the wrong side, OR, the data point is on the correct side of the decision boundary but still closer to it than the desired margin.

$y_i \mathbf{w}^T \mathbf{x}_i < 0$ in Perceptrons means the data point is misclassifed.

(adding $x_i y_i$ to $\mathbf{w}$ so that pushing decision to the right direction when the point is misclassified, is the means of learning in perceptron and SVM)

If $\lambda = 0$, the weight vector is a linear combination of the trainingexamples

$$\mathbf{w}^{(t)} = \sum_{j=1}^{t} \eta^{(j)} y^{(j)} \mathbf{x}^{(j)} \quad if\ \lambda = 0$$

# 4 Dual soft-margin SVM

In simple terms, it refers to a different way of representing and solving the optimization problem that defines the SVM.

The dual representation reformulates the SVM problem in terms of Lagrange multipliers (dual variables), denoted as $\alpha_i$. It focuses on the relationships between data points rather than the hyperplane's characteristics.

In the dual form, the weight vector $\mathbf{w}$ is expressed as a linear combination of the training data points

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

A training data point $\mathbf{x}_i$ is a support vector if and only if $\alpha_i > 0$. For all non-support vectors, $\alpha_i = 0$. This means the optimal hyperplane in an SVM is determined only by the support vectors

Consequently, the functional margin $y \mathbf{w}^T \mathbf{x}$ also can be expressed using the support vectors:

$$y \mathbf{w}^T \mathbf{x} = y \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}$$

The norm of the weight vector can be expressed as

$$\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i^T \sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_j = \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

Note that the training data appears in pairwise inner products: $\mathbf{x}_i^T \mathbf{x}_j$

Kernel function:

$$\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) = \mathbf{x}_i^T \mathbf{x}_j$$

Plug in Margin and Squared norm

$$y\mathbf{w}^T\mathbf{x} = y\sum_{i=1}^{m} \alpha_i y_i \kappa\left(\mathbf{x}_i, \mathbf{x}\right)$$
$$\|\mathbf{w}\|^2 = \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)$$

A dual optimization problem for the soft-margin SVM with kernels is given by

$$\begin{aligned}
&\text{Maximize } OBJ(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right) \\
&\text{w.r.t variables } \boldsymbol{\alpha} \in \mathbb{R}^m \\
&\text{Subject to} \quad 0 \leq \alpha_i \leq C/m \\
&\text{for all } i = 1, \ldots, m
\end{aligned}$$

The kernel function $\kappa\left(\mathbf{x}_i, \mathbf{x}_j\right)$ represents an inner product in a transformed feature space.

This allows the SVM to create non-linear decision boundaries in the original input space by implicitly mapping data points to a higher-dimensional space.

## 4.1 Kernel trick

It allows SVM to operate in a higher-dimensional feature space without explicitly computing the coordinates in that space.

Basis Functions ($\phi$) : map the input data from its original space to a higher-dimensional feature space. $\phi : \mathbb{R}^d \mapsto \mathbb{R}^k$

The transformation aims to convert non-linearly separable data in the original space into linearly separable data in the higher-dimensional space.

A kernel function, denoted as $\kappa_\phi$, computes the inner product of two vectors in the transformed feature space. For two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ in the original space, $\kappa_\phi\left(\mathbf{x}_i, \mathbf{x}_j\right) = \phi\left(\mathbf{x}_i\right)^T \phi\left(\mathbf{x}_j\right)$.

In the transformed space, the optimal hyperplane can be represented as $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \phi(\mathbf{x}_i)$. However, due to the kernel trick, we do not need to compute $\phi(\mathbf{x}_i)$ explicitly.

The decision function in the transformed space becomes $\mathbf{w}^T \phi(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i \kappa_\phi(\mathbf{x}_i, \mathbf{x})$. This allows the classification of new data points using the kernel function.

Similarly, the squared norm of the weight vector in the transformed space is $\|\mathbf{w}\|^2 = \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa_\phi(\mathbf{x}_i, \mathbf{x}_j)$

The kernel trick allows algorithms to operate as if they are in a high-dimensional space, gaining the ability to model complex patterns, without the computational cost of actually working in that space.

## 4.2 Stochastic Dual Coordinate Ascent for dual SVM

SDCA for dual formulation of SVM is a method for solving the optimization problem by iteratively updating one dual variable ($\alpha_i$) at a time.

Coordinate Ascent means updates one dual variable $\alpha_i$ at a time while keeping the other dual variables fixed. This is done iteratively for different variables.

The update is guided by the gradient of the dual SVM objective function OBJ($\alpha$)

$$
\begin{aligned}
&\Delta\alpha_i = \frac{\partial}{\partial \alpha_i} OBJ(\boldsymbol{\alpha}) \\
&= \frac{\partial}{\partial \alpha_i} \left( \sum_{k=1}^{m} \alpha_k - \frac{1}{2} \sum_{k=1}^{m} \sum_{j=1}^{m} \alpha_k \alpha_j y_k y_j \kappa(\mathbf{x}_k, \mathbf{x}_j) \right) \\
&= 1 - y_i \sum_{j=1}^{m} \alpha_j y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \\
&= 1 - y_i f(\mathbf{x}_i)
\end{aligned}
$$

where we used the dual representation

$$
f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} = \sum_{j=1}^{m} \alpha_j y_j \kappa(\mathbf{x}_j, \mathbf{x})
$$

The update direction thus depends on the function margin $y_i f(\mathbf{x}_i)$ :

$$
\Delta\alpha_i = \begin{cases} < 0 & \text{if } y_i f(\mathbf{x}_i) > 1 \\ 0 & \text{if } y_i f(\mathbf{x}_i) = 1 \\ > 0 & \text{if } y_i f(\mathbf{x}_i) < 1 \end{cases}
$$

- If the margin is too small ($y_i f(\mathbf{x}_i) < 1$ ), which means the Hinge loss is positive, $\alpha_i$ is increased.

- If the margin is more than required ($y_i f(\mathbf{x}_i) > 1$ ), $\alpha_i$ is decreased.

- If the margin is exactly $1\,(y_i f(\mathbf{x}_i) = 1)$, $\alpha_i$ remains unchanged.

The role of $\eta$ in the Perceptron (which adjusts the weight vector $\mathbf{w}$ ) is similar to the role of $\alpha_i$ in SDCA (which adjusts the dual variable).

### 4.2.1 Optimal Update Direction and Step-Size

To find the optimal update for $\alpha_i$, you set the partial derivative of the objective function with respect to $\alpha_i$ to zero:

$$\frac{\partial}{\partial \alpha_i}\mathrm{OBJ}(\boldsymbol{\alpha}) = 0 \alpha_i = \frac{1 - y_i \sum_{j \neq i} \alpha_j y_j \kappa(\mathbf{x}_i, \mathbf{x}_j)}{\kappa(\mathbf{x}_i, \mathbf{x}_i)}$$

bound $\alpha_i$ with $0 \leq \alpha_i \leq C/m$ :

$$\alpha_i = \min(C/m, \max(\alpha_i, 0))$$

These bounds are critical as they enforce the constraints of the optimization problem and ensure regularization, controlling the trade-off between maximizing the margin and minimizing the misclassification.

The Algorithmic Process

1. Initialization: Start with all $\alpha_i$ set to zero ($\boldsymbol{\alpha} = \mathbf{0}$). 2. Iterative Updates:

- Randomly select a training example $(x_i, y_i)$.

- Update the corresponding dual variable $\alpha_i$ using the formula derived above.

- Clip $\alpha_i$ to ensure it lies within the prescribed bounds.

3. Repeat until a stopping criterion is satisfied (e.g., a certain number of iterations, convergence criterion, etc.). 4. Return the final set of dual variables $\alpha$.