18-661 Introduction to Machine Learning

Support Vector Machines (SVM) – II

Spring 2024

Announcements

Midterm Exam, 2/28:

- Mix of multiple choice and short answer questions
- Content will include up to SVM (this lecture)
- Topics: MLE/MAP, linear regression; bias-variance tradeoff, overfitting, naive bayes, logistic regression, SVM
- HW3 will (hopefully) be graded before the midterm

More details to follow.

Outline

1. Review: Linear SVM
2. Duality
3. Kernel SVM
4. SVM in Context
5. Summary

Outline

Last Class:

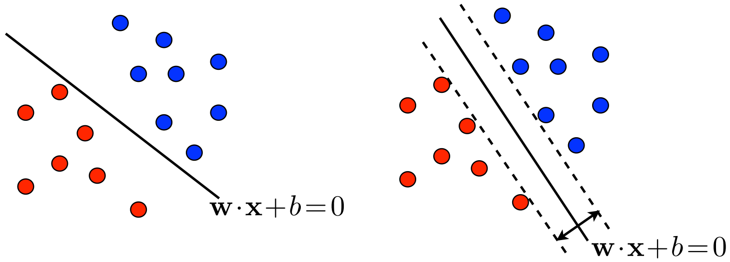
- Max-margin formulation for separable and non-separable SVMs.
- Definition and importance of support vectors.
- Hinge loss formulation of SVMs.
- Equivalence of the max-margin and hinge loss formulations.

Today:

- Duality
- Nonlinear SVM
- SVM in context

Review: Linear SVM

Intuition: Where to Put the Decision Boundary?



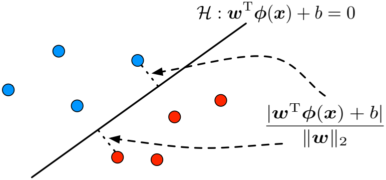
Find a decision boundary in the '*middle*' of the two classes that:

- Perfectly classifies the training data
- Is as far away from every training point as possible

Defining the Margin

Margin
Smallest distance between the hyperplane and all training points

$$\text{MARGIN}(\boldsymbol{w}, b) = \min_n \frac{y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]}{\|\boldsymbol{w}\|_2}$$



How can we use this to find the SVM solution?

Rescaled Margin

We further constrain the problem by scaling $(\boldsymbol{w}, b)$ such that

$$\min_n y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1.$$

which leads to:

$$\textsc{margin}(\boldsymbol{w}, b) = \frac{\min_n y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]}{\|\boldsymbol{w}\|_2} = \frac{1}{\|\boldsymbol{w}\|_2}$$

SVM: Max-margin Formulation for Separable Data

We thus want to solve:

$$\max_{\boldsymbol{w},b} \underbrace{\frac{1}{\|\boldsymbol{w}\|_2}}_{\text{margin}} \quad \text{such that} \quad \underbrace{\min_n y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1}_{\text{scaling of } \boldsymbol{w},\, b}$$

This is equivalent to

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1, \quad \forall \quad n \end{aligned}$$

SVM for Non-separable Data

Constraints in separable setting

$$y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1, \quad \forall \quad n$$

This inherently requires all the training data are correctly separated into two sides of the boundary.

Constraints in non-separable setting
Can we modify our constraints to account for non-separability?

Specifically, we introduce slack variables $\xi_n \geq 0$:

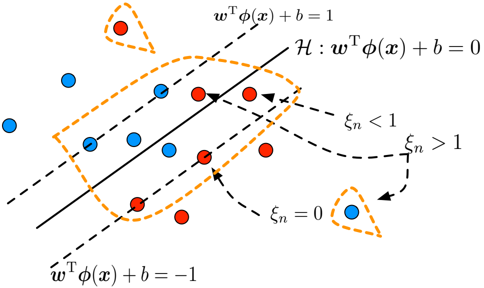
$$y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \quad n$$

Soft-margin SVM Formulation

We do not want ξ_n to grow too large, and we can control their size by incorporating them into our optimization problem:

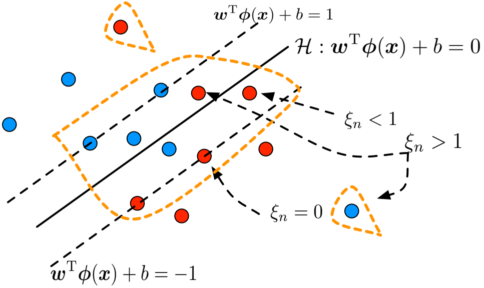
$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$
$$\text{s.t.} \quad y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \; n$$
$$\xi_n \geq 0, \quad \forall \; n$$

Support Vectors: Revisit



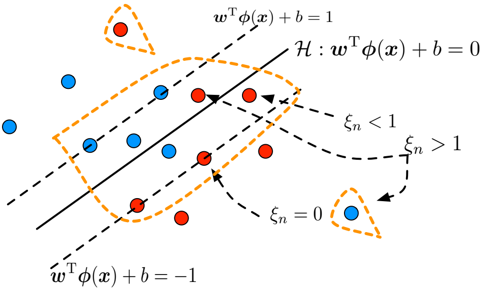
Recall the constraints $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n$ from the soft-margin formulation. All the training points $(\boldsymbol{x}_n, y_n)$ that satisfies the constraint with "$=$" are support vectors.

Support Vectors: Revisit



In other words, support vectors satisfy $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1 - \xi_n$, which can be further divided into several categories:

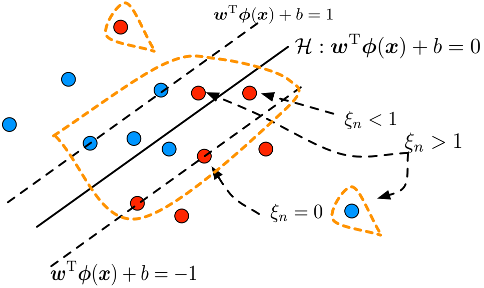
Support Vectors: Revisit



In other words, support vectors satisfy $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1 - \xi_n$, which can be further divided into several categories:

- $\xi_n = 0$: $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$, the point is on the correct side with distance $\frac{1}{\|\boldsymbol{w}\|}$.

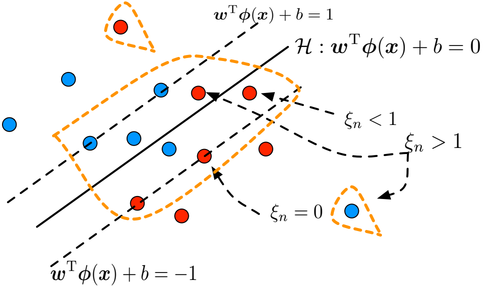
Support Vectors: Revisit



In other words, support vectors satisfy $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1 - \xi_n$, which can be further divided into several categories:

- $\xi_n = 0$: $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$, the point is on the correct side with distance $\frac{1}{\|\boldsymbol{w}\|}$.
- $0 < \xi_n \leq 1$: $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \in [0, 1)$ on the correct side, but with distance less than $\frac{1}{\|\boldsymbol{w}\|}$.

Support Vectors: Revisit



In other words, support vectors satisfy $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1 - \xi_n$, which can be further divided into several categories:

- $\xi_n = 0$: $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$, the point is on the correct side with distance $\frac{1}{\|\boldsymbol{w}\|}$.
- $0 < \xi_n \leq 1$: $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \in [0, 1)$ on the correct side, but with distance less than $\frac{1}{\|\boldsymbol{w}\|}$.
- $\xi_n > 1$: $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] < 0$, on the wrong side of the boundary.

Summary: Three SVM Formulations

In order to learn a linear classifier $\mathbf{y} = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$:

Hard-margin (for separable data)

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 \text{ s.t. } y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1,\ \xi_n \geq 0,\ \forall\ n$$

Soft-margin (add slack variables)

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \text{ s.t. } y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n,\ \xi_n \geq 0,\ \forall\ n$$

Hinge loss (define a loss function for each data point)

$$\min_{\boldsymbol{w},b} \sum_n \max(0, 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Duality

Advantages of SVM

So far, we have shown that SVM is:

1. Is less sensitive to outliers.
2. Maximizes distance of training data from the boundary.
3. Only requires a subset of the training points.
4. Generalizes well to many nonlinear models.
5. Scales better with high-dimensional data.

We will now use duality to show the fourth property.

What is the Lagrangian?

$$\begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_n \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Consider optimization problem

$$p^* = \min f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \preceq 0.$$

What is the Lagrangian?

Consider optimization problem

$$p^* = \min_{x} f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \leq 0.$$

$\{\lambda_n\}$

The Lagrangian is defined as $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \underbrace{\lambda^T \mathbf{g}(\mathbf{x})}_{\sum_n \lambda_n g_n}$.

vec vec

What is the Lagrangian?

Consider optimization problem

$$p^* = \min f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \leq 0.$$

The Lagrangian is defined as $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$.

- λ is called the "Lagrange Multiplier."
- You can think of $\lambda^T \mathbf{g}(\mathbf{x})$ as "penalty" for constraint violation.

What Is Duality?

Consider the optimization problem

$$p^* = \min f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \leq 0.$$

The Lagrangian is defined as $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$.

What Is Duality?

Consider the optimization problem

$$p^* = \min f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \leq 0.$$

The Lagrangian is defined as $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$.

The above (known as primal) is equivalent to $\min_{\mathbf{x}} \max_{\lambda \geq 0} L(\mathbf{x}, \lambda)$.

What Is Duality?

Consider the optimization problem

$$p^* = \min f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \leq 0.$$

≤ 0

The Lagrangian is defined as $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$.

The above (known as primal) is equivalent to $\min_{\mathbf{x}} \underbrace{\max_{\lambda \geq 0}}_{f(x)+0} L(\mathbf{x}, \lambda)$.

- If $g_i(\mathbf{x}) \leq 0$, $\max_{\lambda_i \geq 0} L(\mathbf{x}, \lambda_i) = f(\mathbf{x})$

What Is Duality?

Consider the optimization problem

$$p^* = \min f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \leq 0.$$

The Lagrangian is defined as $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$.

The above (known as primal) is equivalent to $\min_{\mathbf{x}} \max_{\lambda \geq 0} L(\mathbf{x}, \lambda)$.

- If $g_i(\mathbf{x}) \leq 0$, $\max_{\lambda_i \geq 0} L(\mathbf{x}, \lambda_i) = f(\mathbf{x})$
- If $g_i(\mathbf{x}) > 0$, $\max_{\lambda_i \geq 0} L(\mathbf{x}, \lambda_i) = +\infty$

$f(x) + \infty$

What Is Duality?

Consider the optimization problem

$$p^* = \min f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \leq 0.$$

The Lagrangian is defined as $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$.

The above (known as primal) is equivalent to $\min_{\mathbf{x}} \max_{\lambda \geq 0} L(\mathbf{x}, \lambda)$.

- If $g_i(\mathbf{x}) \leq 0$, $\max_{\lambda_i \geq 0} L(\mathbf{x}, \lambda_i) = f(\mathbf{x})$
- If $g_i(\mathbf{x}) > 0$, $\max_{\lambda_i \geq 0} L(\mathbf{x}, \lambda_i) = +\infty$
- Effectively enforces constraint $\mathbf{g}(\mathbf{x}) \leq 0$.

What Is Duality?

Consider the optimization problem

$$p^* = \min f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \leq 0.$$

The Lagrangian is defined as $L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$.

The above (known as primal) is equivalent to $\min_{\mathbf{x}} \max_{\lambda \geq 0} L(\mathbf{x}, \lambda)$.

- If $g_i(\mathbf{x}) \leq 0$, $\max_{\lambda_i \geq 0} L(\mathbf{x}, \lambda_i) = f(\mathbf{x})$
- If $g_i(\mathbf{x}) > 0$, $\max_{\lambda_i \geq 0} L(\mathbf{x}, \lambda_i) = +\infty$
- Effectively enforces constraint $\mathbf{g}(\mathbf{x}) \leq 0$.

Dual problem: swapping the order of min and max

$$d^* = \max_{\lambda \geq 0} \underbrace{\min_x L(x, \lambda)}_{\text{known as dual function } D(\lambda)}$$

Properties of Duality

$$\text{Primal:} \quad p^* = \min_{\mathbf{x}} \max_{\lambda \geq 0} f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$$

$$\text{Dual:} \quad d^* = \max_{\lambda \geq 0} \min_{\mathbf{x}} f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$$

Properties of Duality

$$\text{Primal:} \quad p^* = \min_{\mathbf{x}} \max_{\lambda \geq 0} f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$$

$$\text{Dual:} \quad d^* = \max_{\lambda \geq 0} \min_{\mathbf{x}} f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$$

Strong Duality: $p^* = d^*$ (sometimes)

- The duality gap is the difference $p^* - d^*$
- $p^* - d^* = 0$ under certain conditions (e.g. convex and continuous; discussed further in Convex Optimization)

Properties of Duality

$$\text{Primal:} \quad p^* = \min_{\mathbf{x}} \max_{\lambda \geq 0} f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$$

$$\text{Dual:} \quad d^* = \max_{\lambda \geq 0} \underbrace{\min_{\mathbf{x}} f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})}_{D(\lambda)}$$

Strong Duality: $p^* = d^*$ (sometimes)

- The duality gap is the difference $p^* - d^*$
- $p^* - d^* = 0$ under certain conditions (e.g. convex and continuous; discussed further in Convex Optimization)

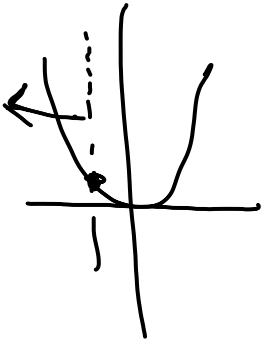
Complementary Slackness: if $p^* = d^*$, then...

- $\mathbf{g}_i(\mathbf{x}) < 0 \Longrightarrow \lambda_i = 0$
- $\lambda_i > 0 \Longrightarrow \mathbf{g}_i(\mathbf{x}) = 0$
- Equivalently, $\lambda_i \mathbf{g}_i(\mathbf{x}) = 0$

Duality: Example

Consider the following problem with optimizer $x^* = -1$, optimal value $\frac{1}{2}$.

$$\min \frac{1}{2}x^2 \text{ s.t. } x + 1 \leq 0$$



Duality: Example

Consider the following problem with optimizer $x^* = -1$, optimal value $\frac{1}{2}$.

$$\min \frac{1}{2}x^2 \text{ s.t. } x + 1 \leq 0$$

Lagrangian $L(x, \lambda) = \frac{1}{2}x^2 + \lambda(x + 1)$

Duality: Example

Consider the following problem with optimizer $x^* = -1$, optimal value $\frac{1}{2}$.

$$\min \frac{1}{2}x^2 \text{ s.t. } x + 1 \leq 0$$

Lagrangian $L(x, \lambda) = \frac{1}{2}x^2 + \lambda(x + 1)$

Dual problem: $D(\lambda) = \min_x L(x, \lambda)$ - how to compute?

- Set $\nabla_x L(x, \lambda) = x + \lambda = 0 \Rightarrow x^*(\lambda) = -\lambda$
- $D(\lambda) = L(x^*(\lambda), \lambda) = -\frac{1}{2}\lambda^2 + \lambda$

$\min_x$

Duality: Example

Consider the following problem with optimizer $x^* = -1$, optimal value $\frac{1}{2}$.

$$\min \frac{1}{2}x^2 \text{ s.t. } x + 1 \leq 0$$

Lagrangian $L(x, \lambda) = \frac{1}{2}x^2 + \lambda(x + 1)$

Dual problem: $D(\lambda) = \min_x L(x, \lambda)$ - how to compute?

- Set $\nabla_x L(x, \lambda) = x + \lambda = 0 \Rightarrow x^*(\lambda) = -\lambda$
- $D(\lambda) = L(x^*(\lambda), \lambda) = -\frac{1}{2}\lambda^2 + \lambda$

Dual solution:

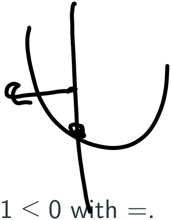
- $\max_{\lambda \geq 0} D(\lambda) = 1/2$ (achieved at $\lambda^* = 1$) — same as the optimal value of the primal
- $x^*(\lambda^*) = -1$ recovers optimal primal solution

$\max_\lambda$

Duality: Example

Recap: for the following problem with optimizer

$$\min \frac{1}{2}x^2 \text{ s.t. } x + 1 \leq 0$$

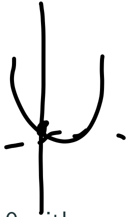


- Primal solution $x^* = -1$ satisfies constraint $x + 1 \leq 0$ with $=$.
- Dual solution $\lambda^* = 1$ is non-zero.

Duality: Example

Recap: for the following problem with optimizer

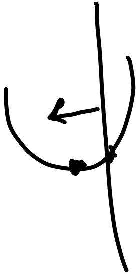
$$\min \frac{1}{2}x^2 \text{ s.t. } x+1 \le 0$$



- Primal solution $x^* = -1$ satisfies constraint $x+1 \le 0$ with $=$.
- Dual solution $\lambda^* = 1$ is non-zero.

Slightly change the problem:

$$\min \frac{1}{2}x^2 \text{ s.t. } x-1 \le 0$$



Duality: Example

Recap: for the following problem with optimizer

$$\min \frac{1}{2}x^2 \text{ s.t. } x + 1 \leq 0$$

- Primal solution $x^* = -1$ satisfies constraint $x + 1 \leq 0$ with $=$.
- Dual solution $\lambda^* = 1$ is non-zero.

Slightly change the problem:

$$\min \frac{1}{2}x^2 \text{ s.t. } x - 1 \leq 0$$

- Primal solution $x^* = 0$ satisfies constraint $x - 1 \leq 0$ with $<$.
- Can show dual solution λ^* is zero.

Duality: Summary

Duality is a way of transforming a constrained optimization problem.

It tells us sometimes-useful information about the problem structure, and can sometimes make the problem easier to solve.

Duality: Summary

Duality is a way of transforming a constrained optimization problem.

It tells us sometimes-useful information about the problem structure, and can sometimes make the problem easier to solve.

- Under strong duality condition, the primal and dual problems are equivalent.
- Further, due to complementary slackness, dual variables tell us whether constraints are met with $=$ or $<$.
- The strong duality condition is not always true for all optimization problems, but is true for the soft-margin SVM problem.

Duality: Summary

Duality is a way of transforming a constrained optimization problem.

It tells us sometimes-useful information about the problem structure, and can sometimes make the problem easier to solve.

- Under strong duality condition, the primal and dual problems are equivalent.
- Further, due to complementary slackness, dual variables tell us whether constraints are met with $=$ or $<$.
- The strong duality condition is not always true for all optimization problems, but is true for the soft-margin SVM problem.

Instead of solving the max margin (primal) formulation, we solve its dual problem which will have certain advantages we will see.

Derivation of the Dual

Here is a skeleton of how to derive the dual problem.

Recipe

1. Formulate the generalized Lagrangian function that incorporates the constraints and introduces dual variables
2. Minimize the Lagrangian function over the primal variables $L(x,\lambda)$
3. Plug in the primal variables from the previous step into the Lagrangian to get the dual function $D(\lambda)$
4. Maximize the dual function with respect to dual variables $\min_x$
5. Recover the solution (for the primal variables) from the dual variables $\max_{\lambda \geq 0}$

Deriving the Dual for SVM

Primal SVM:

margin

$\frac{1}{\|w\|_2}$

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$$\text{s.t.} \quad y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \ \ n$$

$$\xi_n \geq 0, \quad \forall \ n$$

Deriving the Dual for SVM

Primal SVM:

$$\begin{aligned}
\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \\
\text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \;\; n \\
& \xi_n \geq 0, \quad \forall \; n
\end{aligned}$$

The constraints are equivalent to the following canonical forms:

$$-\xi_n \leq 0 \quad \text{and} \quad 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n \leq 0$$

$\gamma_n \geq 0$: $\gamma_n \geq 0$

$\gamma_n \leq 0$

Deriving the Dual for SVM

Primal SVM:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \ n \\ & \xi_n \geq 0, \quad \forall \ n \end{aligned}$$

The constraints are equivalent to the following canonical forms:

$$-\xi_n \leq 0 \quad \text{and} \quad 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n \leq 0$$

Lagrangian:

$$\begin{aligned} L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) = & \ C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n \\ & + \sum_n \alpha_n \{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\} \end{aligned}$$

under the constraints that $\alpha_n \geq 0$ and $\lambda_n \geq 0$.

Deriving the Dual of SVM

Lagrangian

$$L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) = C \sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n$$
$$+ \sum_n \alpha_n \{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

under the constraints that $\alpha_n \geq 0$ and $\lambda_n \geq 0$.

- Primal variables: $\boldsymbol{w}$, b, $\{\xi_n\}$; dual variables $\{\alpha_n\}$, $\{\lambda_n\}$

Deriving the Dual of SVM

Lagrangian

$$L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n\xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

under the constraints that $\alpha_n \geq 0$ and $\lambda_n \geq 0$.

- Primal variables: $\boldsymbol{w}$, b, $\{\xi_n\}$; dual variables $\{\alpha_n\}$, $\{\lambda_n\}$
- Minimize the Lagrangian function over the primal variables by setting $\frac{\partial L}{\partial \boldsymbol{w}} = 0$, $\frac{\partial L}{\partial b} = 0$, and $\frac{\partial L}{\partial \xi_n} = 0$.

Deriving the Dual of SVM

Lagrangian

$$L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) = C \sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n \{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

under the constraints that $\alpha_n \geq 0$ and $\lambda_n \geq 0$.

- Primal variables: $\boldsymbol{w}$, b, $\{\xi_n\}$; dual variables $\{\alpha_n\}$, $\{\lambda_n\}$
- Minimize the Lagrangian function over the primal variables by setting $\frac{\partial L}{\partial \boldsymbol{w}} = 0$, $\frac{\partial L}{\partial b} = 0$, and $\frac{\partial L}{\partial \xi_n} = 0$.
- Substitute primal variables from the above into the Lagrangian to get the dual function.

Deriving the Dual of SVM

Lagrangian

$$L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

under the constraints that $\alpha_n \geq 0$ and $\lambda_n \geq 0$.

- Primal variables: $\boldsymbol{w}$, b, $\{\xi_n\}$; dual variables $\{\alpha_n\}$, $\{\lambda_n\}$
- Minimize the Lagrangian function over the primal variables by setting $\frac{\partial L}{\partial \boldsymbol{w}} = 0$, $\frac{\partial L}{\partial b} = 0$, and $\frac{\partial L}{\partial \xi_n} = 0$.
- Substitute primal variables from the above into the Lagrangian to get the dual function.
- Maximize the dual function with respect to dual variables.

Deriving the Dual of SVM

Looking just at the terms that contain **x**:

$$L(\ldots) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

Deriving the Dual of SVM

Looking just at the terms that contain **x**:

$$L(\ldots) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n \{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

- $\frac{\partial L}{\partial \boldsymbol{w}} = 0 \Longrightarrow \mathbf{w} - \sum_n \alpha_n y_n \mathbf{x}_n = 0 \Longrightarrow \mathbf{w} = \sum_n \alpha_n y_n \mathbf{x}_n$

Deriving the Dual of SVM

Looking just at the terms that contain **x**:

$$L(\ldots) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

- $\dfrac{\partial L}{\partial \boldsymbol{w}} = 0 \Longrightarrow \mathbf{w} - \sum_n \alpha_n y_n \mathbf{x}_n = 0 \Longrightarrow \mathbf{w} = \sum_n \alpha_n y_n \mathbf{x}_n$
- $D(\{\alpha_n\}, \{\lambda_n\}) = \ldots + \frac{1}{2}||w||_2^2 + \sum_n \alpha_n y_n \mathbf{w}^T \mathbf{x}_n + \ldots$

Deriving the Dual of SVM

Looking just at the terms that contain **x**:

$$L(\ldots) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n \{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

- $\frac{\partial L}{\partial \boldsymbol{w}} = 0 \Longrightarrow \mathbf{w} - \sum_n \alpha_n y_n \mathbf{x}_n = 0 \Longrightarrow \mathbf{w} = \sum_n \alpha_n y_n \mathbf{x}_n$
- $D(\{\alpha_n\}, \{\lambda_n\}) = \ldots + \frac{1}{2}||w||_2^2 + \sum_n \overbrace{\alpha_n y_n \mathbf{w}^T \mathbf{x}_n}{} + \ldots$

$$\frac{1}{2}||\mathbf{w}||_2^2 = \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \mathbf{x}_m^T \mathbf{x}_m$$

Deriving the Dual of SVM

Looking just at the terms that contain $\mathbf{x}$:

$$L(\ldots) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

- $\dfrac{\partial L}{\partial \boldsymbol{w}} = 0 \Longrightarrow \mathbf{w} - \sum_n \alpha_n y_n \mathbf{x}_n = 0 \Longrightarrow \mathbf{w} = \sum_n \alpha_n y_n \mathbf{x}_n$
- $D(\{\alpha_n\},\{\lambda_n\}) = \ldots + \dfrac{1}{2}||w||_2^2 + \sum_n \alpha_n y_n \mathbf{w}^T \mathbf{x}_n + \ldots$

$$\frac{1}{2}||\mathbf{w}||_2^2 = \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \mathbf{x}_m^T \mathbf{x}_m$$

$$\sum_n \alpha_n y_n \mathbf{w}^T \mathbf{x}_n = -\sum_n \alpha_n y_n \left(\sum_m \alpha_m y_m \mathbf{x}_m\right)^T \mathbf{x}_n$$

$$= -\sum_{m,n} y_m y_n \alpha_m \alpha_n \mathbf{x}_m^T \mathbf{x}_n$$

$-\frac{1}{2}(\ldots)$

Lagrangian

$$L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) = C \sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n \{\underline{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n}\}$$

If we perform the full procedure, we get the dual function $D(\{\alpha_n\}, \{\lambda_n\})$:

$$\max_{\boldsymbol{\alpha}} \quad \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$$

$1 \leq m \leq N$
$1 \leq n \leq N$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

$$\sum_n \alpha_n y_n = 0$$

Dual Formulation of SVM

Dual is also a convex quadratic program

$$\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n \\
\text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\
& \sum_n \alpha_n y_n = 0
\end{aligned}$$

Dual Formulation of SVM

Dual is also a convex quadratic program

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

- There are N dual variables α_n, one for each data point

Dual Formulation of SVM

Dual is also a convex quadratic program

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n \\ \text{s.t.} \quad & 0 \le \alpha_n \le C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

- There are N dual variables α_n, one for each data point
- Independent of the size d of $\mathbf{x}$: SVM scales better for high-dimensional features.

Dual Formulation of SVM

Dual is also a convex quadratic program

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n \\ \text{s.t.} \quad & 0 \le \alpha_n \le C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0\end{aligned}$$

- There are N dual variables α_n, one for each data point
- Independent of the size d of $\mathbf{x}$: SVM scales better for high-dimensional features.
- May seem like a lot of optimization variables when N is large, but many of the α_n become zero. Why?

Complementary Slackness in SVM

Primal Max-Margin Formulation

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \ n \\ & \xi_n \geq 0, \quad \forall \ n \end{aligned}$$

Dual Formulation

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall \ n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

Why Do Many α_n Become Zero?

Primal Formulation:

(canonical form)

$$\ldots \text{s.t.} \quad y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \quad n \ldots$$

Dual Formulation:

$$\ldots \text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall \quad n \ldots$$

- By complementary slackness:

$$\alpha_n\{1 - \xi_n - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]\} = 0 \quad \forall n$$

Why Do Many α_n Become Zero?

Primal Formulation:

$$\ldots \text{s.t.} \quad y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \quad n \ldots$$

Dual Formulation:

$$\ldots \text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall \quad n \ldots$$

- By complementary slackness:

$$\alpha_n \{ \underline{1 - \xi_n - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]} \} = 0 \quad \forall n$$

- This tells us that $\alpha_n > 0$ only when $1 - \xi_n = y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]$, i.e. (x_n, y_n) is a support vector. So most of the α_n is zero, and the only non-zero α_n are for the support vectors.

Why Do Many α_n Become Zero?

Primal Formulation:

$$\ldots \text{s.t.} \quad y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \quad n \ldots$$

Dual Formulation:

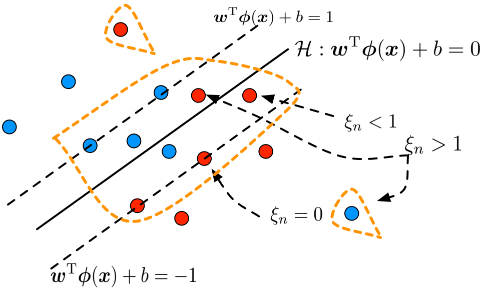
$$\ldots \text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall \quad n \ldots$$

- By complementary slackness:

$$\alpha_n\{1 - \xi_n - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]\} = 0 \quad \forall n$$

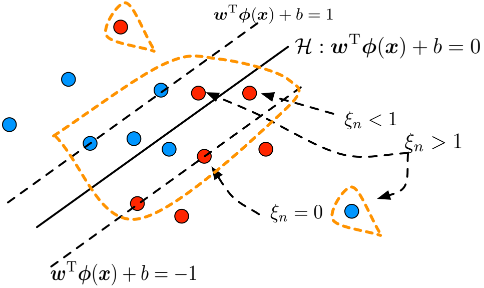
- This tells us that $\alpha_n > 0$ only when $1 - \xi_n = y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]$, i.e. (x_n, y_n) is a support vector. So most of the α_n is zero, and the only non-zero α_n are for the support vectors.
- Further, $\alpha_n < C$ only when $\xi_n = 0$. (The derivation of this is beyond the scope of today's lecture)

Visualizing the Support Vectors



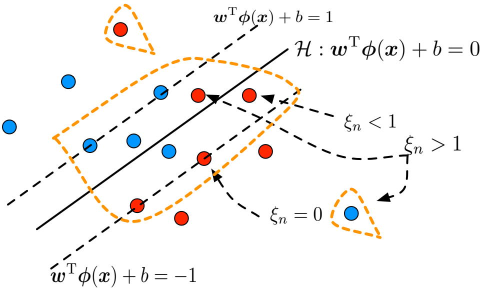
- $\alpha_n = 0 \Longrightarrow \xi_n = 0, y_n[\mathbf{w}^T\mathbf{x}_n + b] \geq 1$: non-support vector (with some edge cases).

Visualizing the Support Vectors



- $\alpha_n = 0 \Longrightarrow \xi_n = 0, y_n[\mathbf{w}^T\mathbf{x}_n + b] \geq 1$: non-support vector (with some edge cases).
- $0 < \alpha_n < C \Longrightarrow \xi_n = 0, y_n[\mathbf{w}^T\mathbf{x}_n + b] = 1$: support vector with distance to boundary $\frac{1}{\|w\|}$.

Visualizing the Support Vectors



- $\alpha_n = 0 \implies \xi_n = 0, y_n[\mathbf{w}^T\mathbf{x}_n + b] \geq 1$: non-support vector (with some edge cases).
- $0 < \alpha_n < C \implies \xi_n = 0, y_n[\mathbf{w}^T\mathbf{x}_n + b] = 1$: support vector with distance to boundary $\frac{1}{\|w\|}$.
- $\alpha_n = C \implies \xi_n < 0, y_n[\boldsymbol{w}^\top\boldsymbol{x}_n + b] < 1$: support vector which violates the margin.

How to Get w and b?

Lagrangian:

$$L(\ldots) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

How to Get w and b?

Lagrangian:

$$L(\ldots) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n \{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

only $\alpha_n > 0$

Recovering **w**:

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \Longrightarrow \boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{x}_n$$

terms contribute

Only depends on support vectors, i.e., points with $\alpha_n > 0$!

How to Get w and b?

Lagrangian:

$$L(\ldots) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n\}$$

Recovering **w**:

$$\frac{\partial L}{\partial \boldsymbol{w}} = 0 \Longrightarrow \boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{x}_n$$

Only depends on support vectors, i.e., points with $\alpha_n > 0$!

Recovering b:

If you can find a sample (x_n, y_n) such that $0 < \alpha_n < C$, (more complicated if you can't), use $y_n \in \{-1, 1\}$:

$$y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$$

$$b = y_n - \boldsymbol{w}^\top \boldsymbol{x}_n = y_n - \sum_m \alpha_m y_m \boldsymbol{x}_m^\top \boldsymbol{x}_n$$

Summary of Dual Formulation

Primal Max-Margin Formulation

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$$\text{s.t.} \quad y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \; n$$

$$\xi_n \geq 0, \quad \forall \; n$$

Dual Formulation

$$\max_{\boldsymbol{\alpha}} \quad \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall \; n$$

$$\sum_n \alpha_n y_n = 0$$

- In dual formulation, the # of variables is independent of dimension.
- Most of the dual variables are 0, and the non-zero ones are the support vectors.
- Can easily recover the primal solution $\boldsymbol{w}, b$ from dual solution.

Advantages of SVM

We have shown that SVM:

1. Maximizes distance of training data from the boundary
2. Only requires a subset of the training points.
3. Is less sensitive to outliers.
4. Scales better with high-dimensional data.
5. Generalizes well to many nonlinear models.

Advantages of SVM

We have shown that SVM:

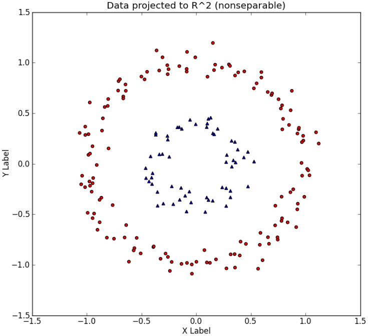
1. Maximizes distance of training data from the boundary
2. Only requires a subset of the training points.
3. Is less sensitive to outliers.
4. Scales better with high-dimensional data.
5. Generalizes well to many nonlinear models.

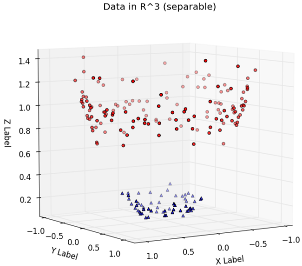
Next: nonlinearity using the "Kernel Trick."

Kernel SVM

Naive nonlinearity

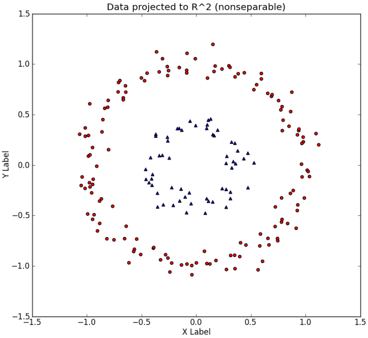
What if the data is not linearly separable?

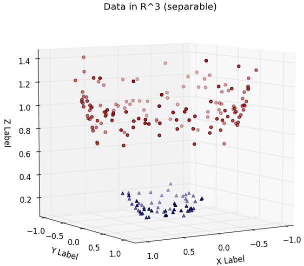




Naive nonlinearity

What if the data is not linearly separable?





Do some feature engineering, e.g. use a feature transformation $\phi(x) = [x_1, x_2, x_1^2 + x_2^2]$ to transform the data in a 3D space.

Can we do better?

Feature engineering is very labor-intensive, and doesn't scale ...

Can we do better?

Feature engineering is very labor-intensive, and doesn't scale ...

What if we can automatically pick features instead?

- Toss in every feature transformation we can think of?
- Randomly generate nonlinear projections of the data?

Can we do better?

Feature engineering is very labor-intensive, and doesn't scale ...

What if we can automatically pick features instead?

- Toss in every feature transformation we can think of?
- Randomly generate nonlinear projections of the data?

We can do even better!

Key insight: the dual problem does not depend on $\mathbf{x}$ or $\phi(\mathbf{x})$, only $\phi(\mathbf{x})^T\phi(\mathbf{x})$.

Primal and Dual SVM Formulations: Kernel Versions

Primal:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$x \to \phi(x) \to$

$$\text{s.t.} \quad y_n[\underline{\boldsymbol{w}^\top \phi(\boldsymbol{x}_n)} + b] \geq 1 - \xi_n, \quad \forall \ n$$

$$\xi_n \geq 0, \quad \forall \ n$$

Dual:

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \underline{\phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)}$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall \ n$$

$$\sum_n \alpha_n y_n = 0$$

Primal and Dual SVM Formulations: Kernel Versions

Primal:

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$$\text{s.t.} \quad y_n[\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b] \geq 1 - \xi_n, \quad \forall \;\; n$$

$$\xi_n \geq 0, \quad \forall \; n$$

Dual:

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall \; n$$

$$\sum_n \alpha_n y_n = 0$$

- In the dual problem, we only need $\phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$.
- ϕ can be very complicated, even infinite dimensional, as long as we know how to calculate $\phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$.

The Kernel Trick

We replace the inner products $\phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$ with a kernel function

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\
\text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\
& \sum_n \alpha_n y_n = 0
\end{aligned}
$$

The Kernel Trick

We replace the inner products $\phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$ with a kernel function

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\ \text{s.t.} \quad & 0 \le \alpha_n \le C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

$$K = \begin{array}{cc} \left[\begin{array}{ccc} & \vdots & \\ \cdots & \cdot & \cdots \\ & \vdots & \end{array}\right] & m \\ n & \end{array} \quad k(x_m, x_n)$$

What is kernel function?

- $k(\mathbf{x}_m, \mathbf{x}_n)$ is a scalar valued function that measures the similarity of $\mathbf{x}_m$ and $\mathbf{x}_n$
- $k(\mathbf{x}_m, \mathbf{x}_n)$ is a valid kernel function if it is symmetric and positive-definite.

The Kernel Trick

We replace the inner products $\phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$ with a kernel function

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\
\text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\
& \sum_n \alpha_n y_n = 0
\end{aligned}
$$

The Kernel Trick

We replace the inner products $\boldsymbol{\phi}(\boldsymbol{x}_m)^\top \boldsymbol{\phi}(\boldsymbol{x}_n)$ with a kernel function

$$
\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\
\text{s.t.} \quad & 0 \le \alpha_n \le C, \quad \forall\ n \\
& \sum_n \alpha_n y_n = 0
\end{aligned}
$$

Why we can use kernel function to replace $\boldsymbol{\phi}(\boldsymbol{x}_m)^\top \boldsymbol{\phi}(\boldsymbol{x}_n)$? Each valid kernel $k(\mathbf{x}_m, \mathbf{x}_n)$ will implicitly define a $\phi(\mathbf{x})$ in the sense $k(\mathbf{x}_m, \mathbf{x}_n) = \phi(\mathbf{x}_m)^\top \phi(\mathbf{x}_n)$.

Note that we don't have to compute ϕ or even need to know *how* to compute it!

Examples of Popular Kernel Functions

Here are some example kernel functions and the corresponding feature.

- Dot product:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{x}_n, \text{ corresponding } \phi(\mathbf{x}) = \mathbf{x}$$

Examples of Popular Kernel Functions

Here are some example kernel functions and the corresponding feature.

- Dot product:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{x}_n, \text{ corresponding } \phi(\mathbf{x}) = \mathbf{x}$$

- Dot product with PD matrix $\mathbf{Q}$:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{Q} \mathbf{x}_n, \text{ corresponding } \phi(\mathbf{x}) = \mathbf{Q}^{1/2} \mathbf{x}$$

Examples of Popular Kernel Functions

Here are some example kernel functions and the corresponding feature.

- Dot product:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{x}_n, \text{ corresponding } \phi(\mathbf{x}) = \mathbf{x}$$

- Dot product with PD matrix $\mathbf{Q}$:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{Q} \mathbf{x}_n, \text{ corresponding } \phi(\mathbf{x}) = \mathbf{Q}^{1/2}\mathbf{x}$$

- Polynomial kernels (corresponding $\phi(\mathbf{x})$ complicated):

$$k(\mathbf{x}_m, \mathbf{x}_n) = (1 + \mathbf{x}_m^\top \mathbf{x}_n)^d, \quad d \in \mathbb{Z}^+$$

Examples of Popular Kernel Functions

Here are some example kernel functions and the corresponding feature.

- Dot product:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{x}_n, \text{ corresponding } \phi(\mathbf{x}) = \mathbf{x}$$

- Dot product with PD matrix $\mathbf{Q}$:

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{Q} \mathbf{x}_n, \text{ corresponding } \phi(\mathbf{x}) = \mathbf{Q}^{1/2}\mathbf{x}$$

- Polynomial kernels (corresponding $\phi(\mathbf{x})$ complicated):

$$k(\mathbf{x}_m, \mathbf{x}_n) = (1 + \mathbf{x}_m^\top \mathbf{x}_n)^d, \quad d \in \mathbb{Z}^+$$

$\exp(-\gamma\Delta)$

$\frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2}\frac{\Delta^2}{\sigma^2})$

- Radial basis kernel (corresponding $\phi(\mathbf{x})$ complicated):

$$k(\mathbf{x}_m, \mathbf{x}_n) = \exp\left(-\gamma \underbrace{\|\mathbf{x}_m - \mathbf{x}_n\|}_{\Delta}^2\right) \text{ for some } \gamma > 0$$

and many more.

Test Prediction

Learning w and b:

$z^{\top}x$

$\phi(x)^{\top}\phi(z)$

$$\boldsymbol{w} = \sum_n \alpha_n y_n \phi(\boldsymbol{x}_n),$$

$$b = y_n - \boldsymbol{w}^{\top}\phi(\boldsymbol{x}_n) = y_n - \sum_m \alpha_m y_m k(\boldsymbol{x}_m, \boldsymbol{x}_n)$$

But for test prediction on a new point $\mathbf{x}$, do we need the form of $\phi(\mathbf{x})$ in order to find the sign of $\boldsymbol{w}^{\top}\phi(\boldsymbol{x}) + b$?

Learning w and b:

$$w = \sum_n \alpha_n y_n \phi(x_n),$$

$$b = y_n - w^\top \phi(x_n) = y_n - \sum_m \alpha_m y_m k(x_m, x_n)$$

But for test prediction on a new point $\mathbf{x}$, do we need the form of $\phi(\mathbf{x})$ in order to find the sign of $w^\top \phi(x) + b$? Fortunately, no!

Test Prediction:

$$h(x) = \text{SIGN}(\sum_n y_n \alpha_n k(x_n, x) + b)$$

$\alpha_n > 0$ only if x_n is a S.V.

At test time it suffices to know the kernel function! So we really do not need to know ϕ.

Summary of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Select a kernel. In general, you don't need to concretely define $\phi(\mathbf{x})$ and can just use one of the popular kernel functions (polynomial kernel or radial kernel).

Summary of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Select a kernel. In general, you don't need to concretely define $\phi(\mathbf{x})$ and can just use one of the popular kernel functions (polynomial kernel or radial kernel).

Training

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n k(\mathbf{x}_m, \mathbf{x}_m)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

$$\sum_n \alpha_n y_n = 0$$

Summary of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Select a kernel. In general, you don't need to concretely define $\phi(\mathbf{x})$ and can just use one of the popular kernel functions (polynomial kernel or radial kernel).

Training

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n k(\mathbf{x}_m, \mathbf{x}_m)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

$$\sum_n \alpha_n y_n = 0$$

Prediction

$$h(\boldsymbol{x}) = \text{SIGN}\left(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b\right)$$

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

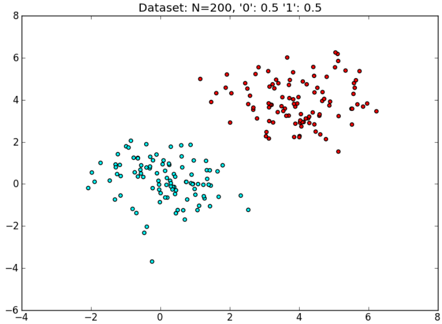


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Here is the decision boundary with linear soft-margin SVM

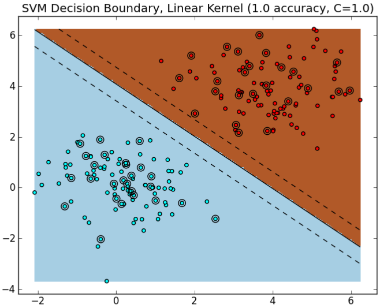


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

What if the data is not linearly separable?

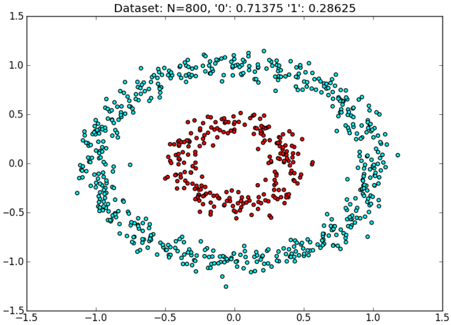


Image Source: https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

The linear decision boundary is pretty bad...

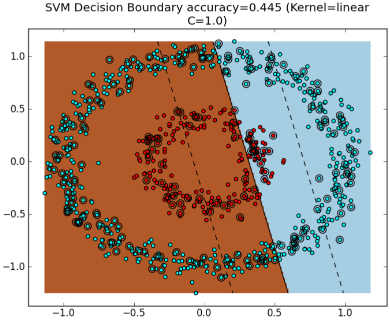
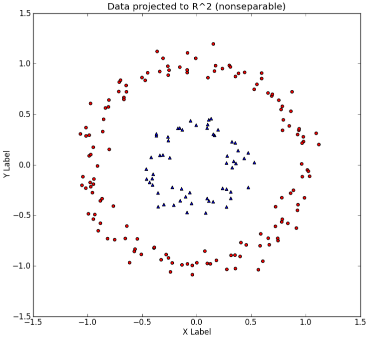


Image Source: https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Use feature $\phi(x) = [x_1, x_2, x_1^2 + x_2^2]$ to transform the data in a 3D space



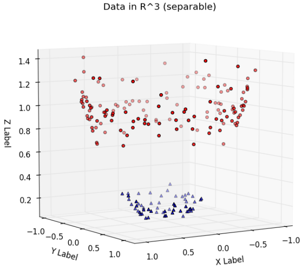


Image Source: https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Then find the decision boundary. How? Solve the dual problem!

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^\top \boldsymbol{\phi}(\boldsymbol{x}_n)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

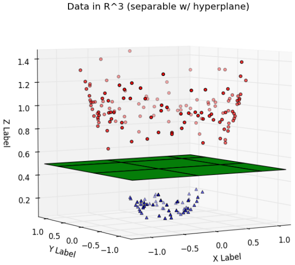
$$\sum_n \alpha_n y_n = 0$$

Then find $\mathbf{w}$ and b. Predict $y = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}) + b)$.

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Here is the resulting decision boundary



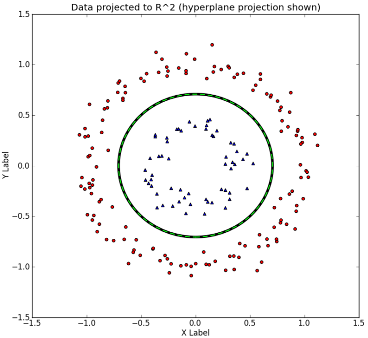


Image Source: https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Effect of the choice of kernel: Polynomial kernel (degree 4)

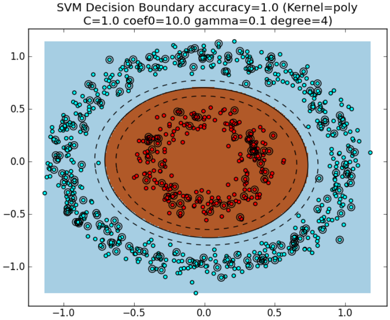


Image Source: https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Effect of the choice of kernel: Radial Basis Kernel

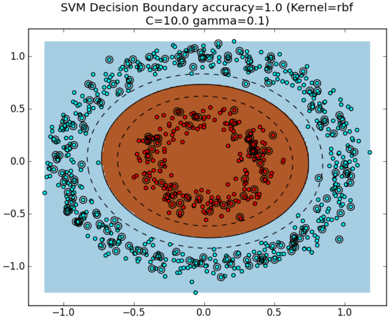


Image Source: https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

Advantages of SVM

Now we have shown all of the below.

1. Maximizes distance of training data from the boundary
2. Only requires a subset of the training points.
3. Is less sensitive to outliers.
4. Scales better with high-dimensional data.
5. Generalizes well to many nonlinear models.

SVM in Context

Linear vs Kernel SVM

If the data is not linearly separable, should we use Linear or Kernel SVM if the data...

- can't be easily transformed to be linearly separable?

Linear vs Kernel SVM

If the data is not linearly separable, should we use Linear or Kernel SVM if the data...

- can't be easily transformed to be linearly separable? Kernel
- can be transformed to be linearly separable?

Linear vs Kernel SVM

If the data is not linearly separable, should we use Linear or Kernel SVM if the data...

- can't be easily transformed to be linearly separable?
- can be transformed to be linearly separable?
- can be transformed into a high dimensional space to be linearly separable?

Linear vs Kernel SVM

If the data is not linearly separable, should we use Linear or Kernel SVM if the data...

- can't be easily transformed to be linearly separable?
- can be transformed to be linearly separable?
- can be transformed into a high dimensional space to be linearly separable?
- can be transformed into a low dimensional space to linearly separable?

Linear vs Kernel SVM

If the data is not linearly separable, should we use Linear or Kernel SVM if the data...

- can't be easily transformed to be linearly separable?
- can be transformed to be linearly separable?
- can be transformed into a high dimensional space to be linearly separable?
- can be transformed into a low dimensional space to linearly separable?

If the data is linearly separable, does it still make sense to use Kernel SVM?

What does SVM address?

SVM addresses:

- Scaling with dataset size (via sparse support vectors)
- Scaling with high dimensionality (via dual problem)
- "Nonparametric" nonlinearity

What does SVM address?

SVM addresses:

- Scaling with dataset size (via sparse support vectors)
- Scaling with high dimensionality (via dual problem)
- "Nonparametric" nonlinearity

SVM does not address:

- Scaling with dataset size and high dimensionality
- Main optimization loop of primal SVM is O(dimensionality)
- Dual SVM is O(dataset size)

— K is O(dataset size ^{2})

What happened to SVMs?

Trees!

You will see that tree ensemble methods can be (circa 2000, e.g. random forest) can scale well with high-dimensionality, dataset size, while handling nonlinearity.

What happened to SVMs?

Some attempt to maintain relevance:

- As data systems scaled, dataset size became much more important than dimensionality

What happened to SVMs?

Some attempt to maintain relevance:

- As data systems scaled, dataset size became much more important than dimensionality
- What if we used linear SVM, but randomly generated $\phi(\mathbf{x})$ so that $\phi(\mathbf{x}_m)^T\phi(\mathbf{x}_n) = k(\mathbf{x}_m, \mathbf{x}_n)$ for a common k?
- "Random Features for Large-Scale Kernel Machines," Rahimi & Recht (2007)

What are SVMs still used for?

$$\text{sign}(w^T x + b)$$

Thoughts?

- Small data, simple problem
- Very efficient

What are SVMs still used for?

Thoughts?

- Strict inference-time compute requirements
- Explainability, e.g. regulatory or liability reasons
- Low-dimensional data
- If a very simple model is sufficient

Summary

Three SVM Formulations

Hard-margin (for separable data)

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 \text{ s.t. } y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1, \ \xi_n \geq 0, \ \forall \ n$$

Soft-margin (add slack variables)

Quadratic Solver

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \ \text{ s.t. } y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \ \xi_n \geq 0, \ \forall \ n$$

Hinge loss (define a loss function for each data point)

$$\min_{\boldsymbol{w},b} \ \sum_n \max(0, 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Dual Formulation

Primal Max-Margin Formulation

$$\begin{aligned} \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C \sum_n \xi_n \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \; n \\ & \xi_n \geq 0, \quad \forall \; n \end{aligned}$$

Dual Formulation

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall \; n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

Select a kernel. In general, you don't need to concretely define $\phi(\mathbf{x})$ and can just use one of the popular kernel functions (polynomial kernel or radial kernel).

Kernel SVM

Select a kernel. In general, you don't need to concretely define $\phi(\mathbf{x})$ and can just use one of the popular kernel functions (polynomial kernel or radial kernel).

Training

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n k(\mathbf{x}_m, \mathbf{x}_m)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

$$\sum_n \alpha_n y_n = 0$$

Prediction

$$h(\boldsymbol{x}) = \text{SIGN}(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b)$$

Advantages of SVM

We have now seen why SVM:

1. Is less sensitive to outliers.
2. Maximizes distance of training data from the boundary.
3. Only requires a subset of the training points.
4. Generalizes well to many nonlinear models.
5. Scales better with high-dimensional data.

Summary

You should know:

- Max-margin formulation for separable and non-separable SVMs.
- Definition and importance of support vectors.
- Hinge loss formulation of SVMs.
- Equivalence of the max-margin and hinge loss formulations.
- Complementary slackness and strong duality in SVM.
- Dual vs Primal SVM.
- Kernel SVMs and the Kernel Trick.