EE4077 Fundamentals of Machine Learning

Support Vector Machines

Fall 2021

EE–Marmara University

Outline

1. Support Vector Machines (SVM): Intuition
2. SVM: Max Margin Formulation
3. SVM: Hinge Loss Formulation
4. Equivalence of These Two Formulations
5. Lagrange Duality and KKT Conditions
6. Dual Derivation of SVMs
7. Kernel SVM

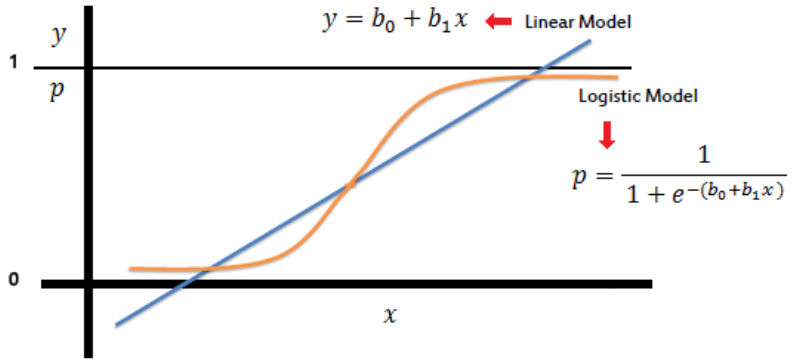
Support Vector Machines (SVM): Intuition

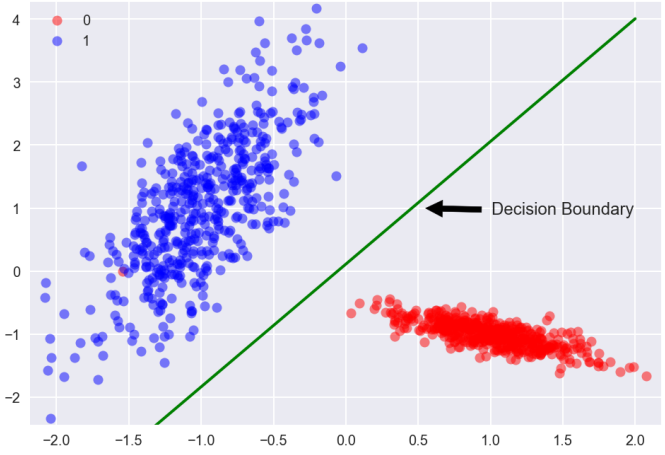
Why do we need SVM?

Alternative to Logistic Regression and Naive Bayes.

- Logistic regression and naive Bayes train over the whole dataset.
- These can require a lot of memory in high-dimensional settings.
- SVM can give a better and more efficient solution
- SVM is one of the most powerful and commonly used ML algorithms

Binary logistic regression





- We only need to know if $p(\boldsymbol{x}) > 0.5$ or < 0.5.
- We don't (always) need to know how far $\boldsymbol{x}$ is from this boundary.

How can we use this insight to improve the classification algorithm?

- What if we just looked at the boundary?
- Maybe then we could ignore some of the samples?

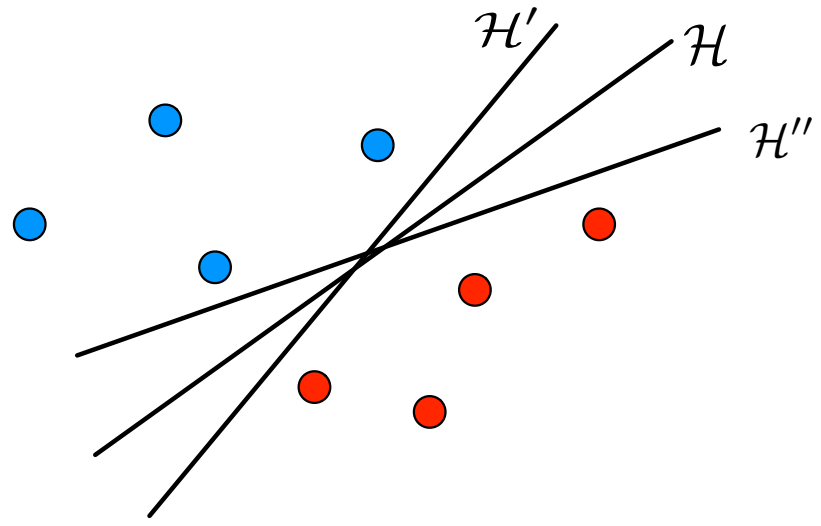
Advantages of SVM

We will see later that SVM:

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

SVM: Max Margin Formulation

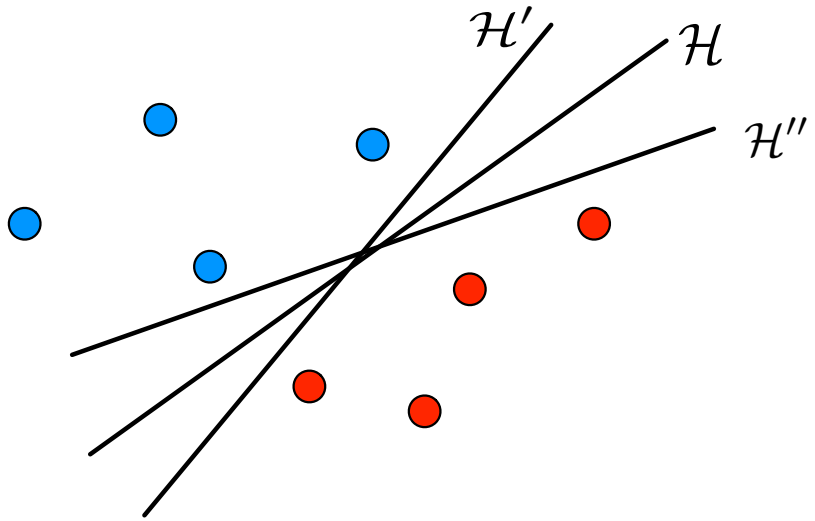
Binary Classification: Finding a Linear Decision Boundary



- Input features $\boldsymbol{x}$.
- Decision boundary is a hyperplane $\mathcal{H} : \boldsymbol{w}^\top \boldsymbol{x} + b = 0$.
- All $\boldsymbol{x}$ satisfying $\boldsymbol{w}^\top \boldsymbol{x} + b < 0$ lie on the same side of the line and are in the same "class."

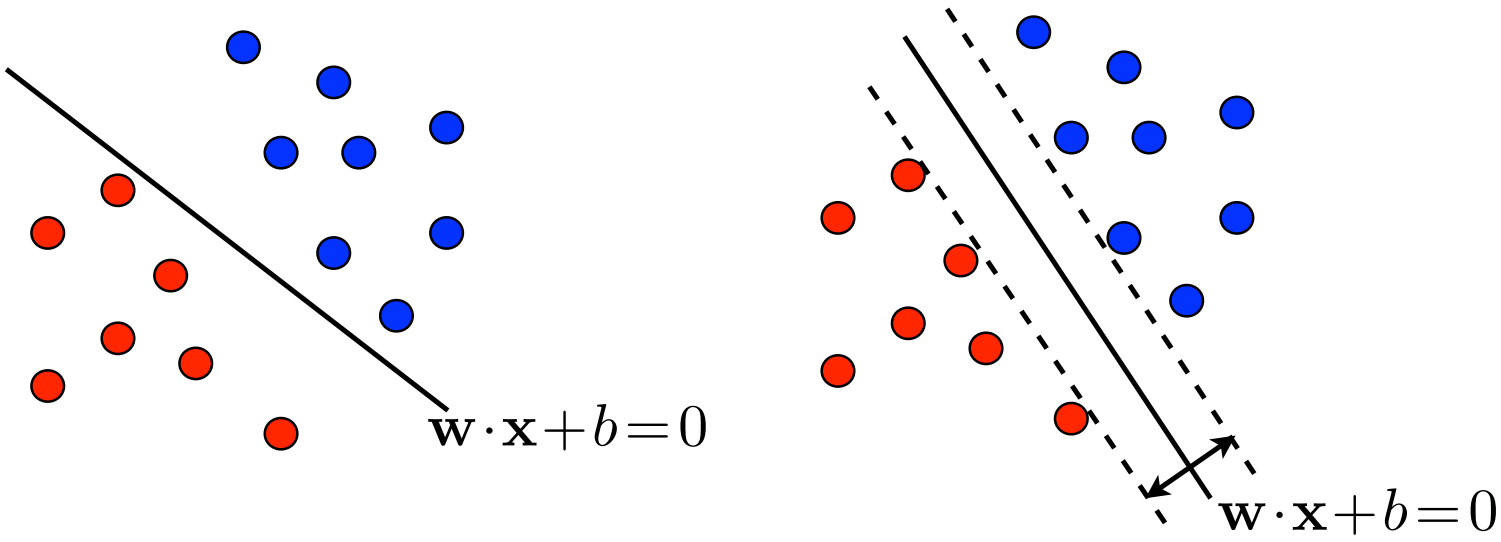
Intuition: Where to put the decision boundary?

- Consider a *separable* training dataset (e.g., with two features)
- There are an infinite number of decision boundaries $\mathcal{H} : \boldsymbol{w}^\top \boldsymbol{x} + b = 0$!



- Which one should we pick?

Intuition: Where to put the decision boundary?



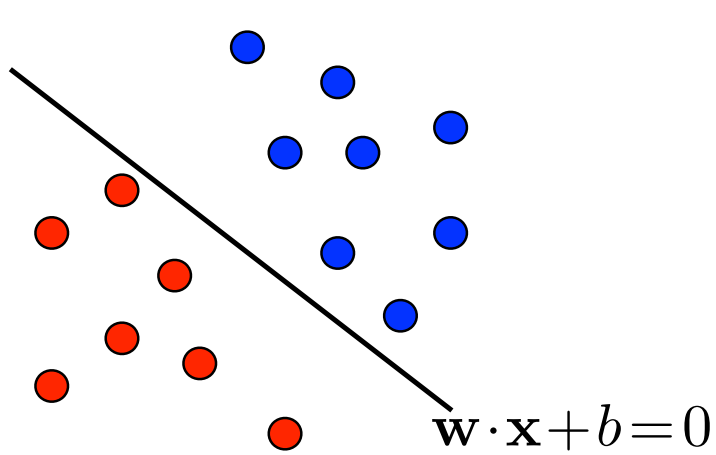
Idea: Find a decision boundary in the '*middle*' of the two classes that:

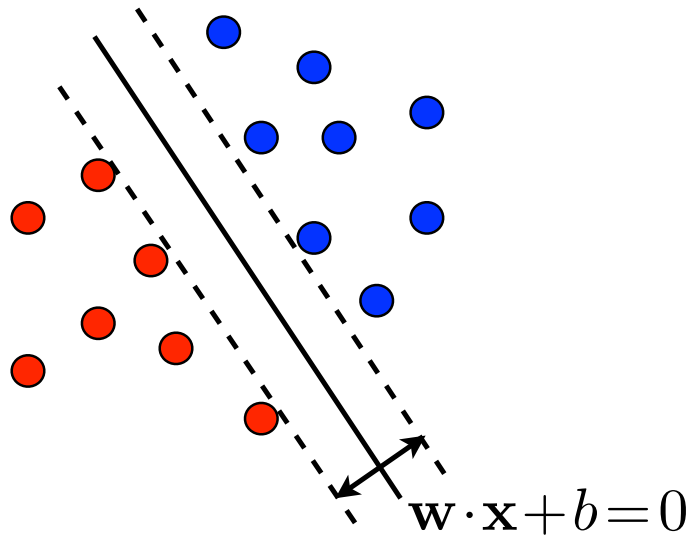
- Perfectly classifies the training data
- Is as far away from every training point as possible

Let us apply this intuition to build a classifier that MAXIMIZES THE MARGIN between training points and the decision boundary

First, we need to review some vector geometry

What is a hyperplane?

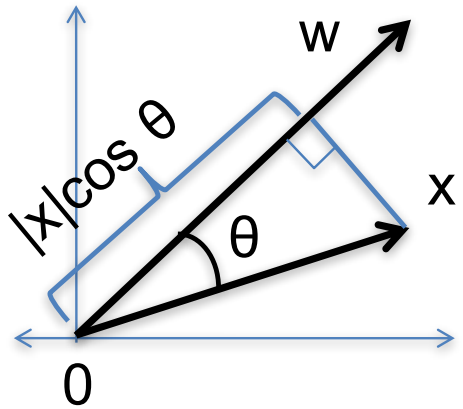




- General equation is $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$
- Divides the space in half, i.e., $\boldsymbol{w}^\top \boldsymbol{x} + b > 0$ and $\boldsymbol{w}^\top \boldsymbol{x} + b < 0$
- A hyperplane is a line in 2D and a plane in 3D
- $\boldsymbol{w} \in \mathbb{R}^d$ is a non-zero normal vector

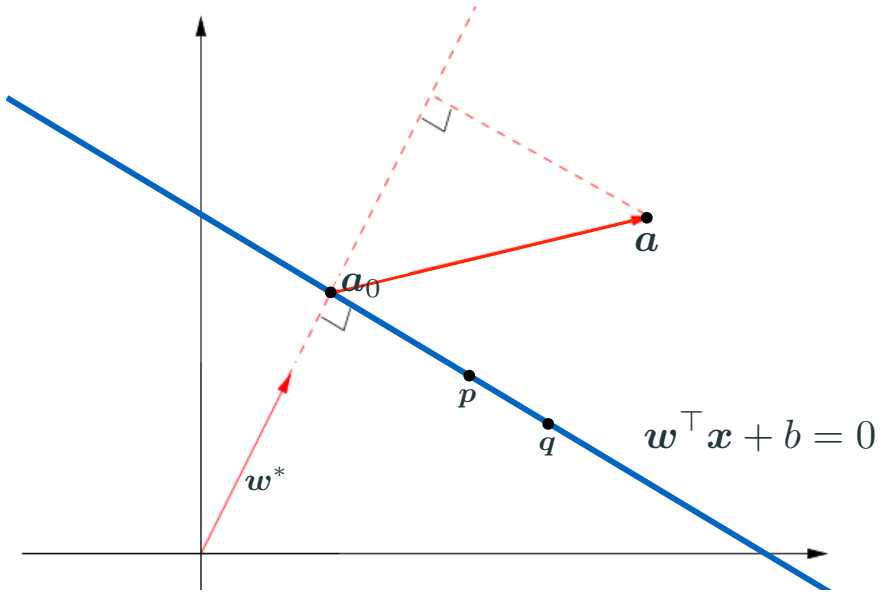
Vector Norms and Inner Products

- Given two vectors $\mathbf{w}$ and $\mathbf{x}$, what is their inner product?
- Inner Product $\mathbf{w}^\top \mathbf{x} = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$



- Inner Product $\mathbf{w}\top\mathbf{x}$ is also equal to $||\boldsymbol{w}||||\boldsymbol{x}||\cos\theta$
- $\mathbf{w}\top\mathbf{w} = ||\boldsymbol{w}||^2$
- If $\mathbf{w}$ and $\mathbf{x}$ are perpendicular $\theta = \pi/2$, and thus the inner product is zero

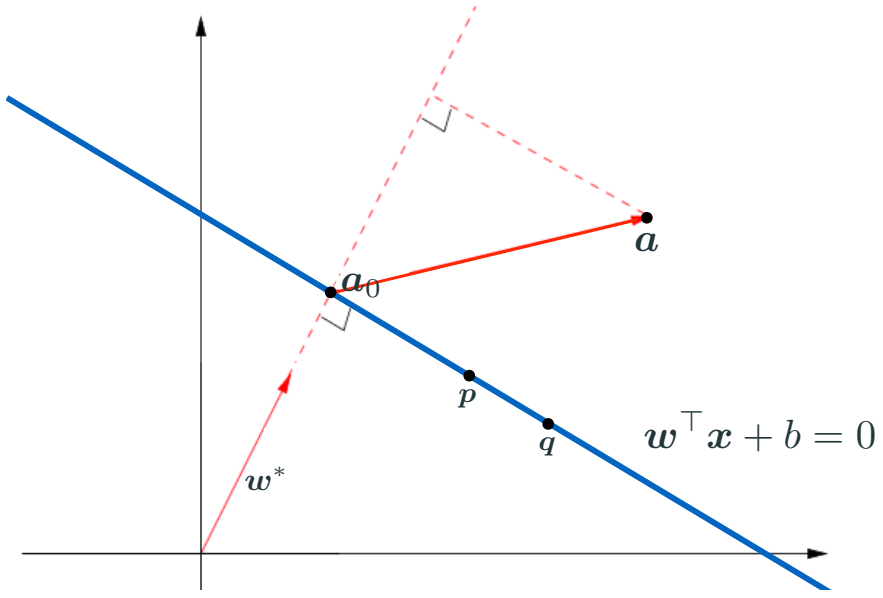
Normal vector of a hyperplane



Vector w is normal to the hyperplane. Why?

- If p and q are both on the line, then $w^\top p + b = w^\top q + b = 0$.
- Then $w^\top (p - q) = w^\top p - w^\top q = -b - (-b) = 0$
- $p - q$ is an arbitrary vector parallel to the line, thus w is orthogonal
- $w^* = \frac{w}{||w||}$ is the unit normal vector

Distance from a Hyperplane



How to find the distance from a to the hyperplane?

- We want to find distance between $\boldsymbol{a}$ and line in the direction of $\boldsymbol{w}^*$.
- If we define point $\boldsymbol{a}_0$ on the line, then this distance corresponds to length of $\boldsymbol{a} - \boldsymbol{a}_0$ in direction of $\boldsymbol{w}^*$, which equals $\boldsymbol{w}^{*\top}(\boldsymbol{a} - \boldsymbol{a}_0)$
- We know $\boldsymbol{w}^\top \boldsymbol{a}_0 = -b$ since $\boldsymbol{w}^\top \boldsymbol{a}_0 + b = 0$.
- Then the distance equals $\frac{1}{||\boldsymbol{w}||}(\boldsymbol{w}^\top \boldsymbol{a} + b)$

Distance from a point to decision boundary

The *unsigned* distance from a point $\boldsymbol{x}$ to decision boundary (hyperplane) $\mathcal{H}$ is

$$d_{\mathcal{H}}(\boldsymbol{x}) = \frac{|\boldsymbol{w}^\top \boldsymbol{x} + b|}{\|\boldsymbol{w}\|_2}$$

We can remove the absolute value $|\cdot|$ by exploiting the fact that the decision boundary classifies every point in the training dataset correctly.

Namely, $(\boldsymbol{w}^\top \boldsymbol{x} + b)$ and $\boldsymbol{x}$'s label y must have the same sign, so:

$$d_{\mathcal{H}}(\boldsymbol{x}) = \frac{y[\boldsymbol{w}^\top \boldsymbol{x} + b]}{\|\boldsymbol{w}\|_2}$$

Notation change from Logistic Regression

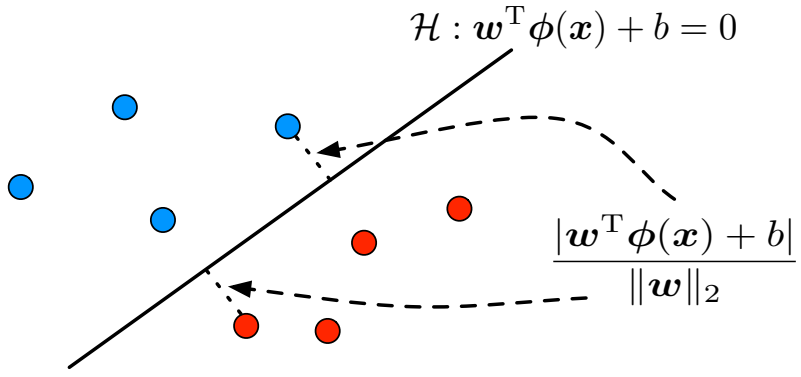
- Change of notation $y = 0 \rightarrow y = -1$
- Separate the bias term b from $\mathbf{w}$

Defining the Margin

Margin

Smallest distance between the hyperplane and all training points

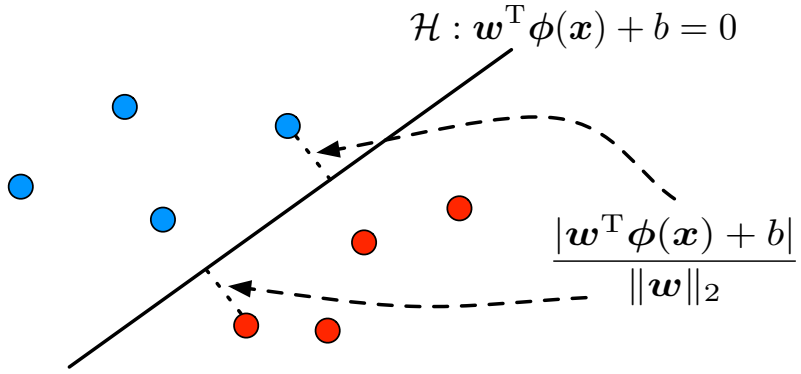
$$\text{MARGIN}(\boldsymbol{w}, b) = \min_n \frac{y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]}{\|\boldsymbol{w}\|_2}$$



Notation change from Logistic Regression

- Change of notation $y = 0 \rightarrow y = -1$
- Separate the bias term b from $\mathbf{w}$

Optimizing the Margin



How should we pick $(\boldsymbol{w}, b)$ based on its margin?

We want a decision boundary that is as far away from all training points as possible, so we to *maximize* the margin!

$$\max_{\boldsymbol{w},b} \left(\min_{n} \frac{y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]}{\|\boldsymbol{w}\|_2} \right) = \max_{\boldsymbol{w},b} \left(\frac{1}{\|\boldsymbol{w}\|_2} \min_{n} y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \right)$$

Only involves points near the boundary (more on this later).

Scale of w

Margin

Smallest distance between the hyperplane and all training points

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

Consider three hyperplanes

$$(\boldsymbol{w}, b) \quad (2\boldsymbol{w}, 2b) \quad (.5\boldsymbol{w}, .5b)$$

Which one has the largest margin?

- The MARGIN doesn't change if we scale $(\boldsymbol{w}, b)$ by a constant c
- $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ and $(c\boldsymbol{w})^\top \boldsymbol{x} + (cb) = 0$: same decision boundary!
- Can we further constrain the problem so as to get a unique solution $(\boldsymbol{w}, b)$?

Rescaled Margin

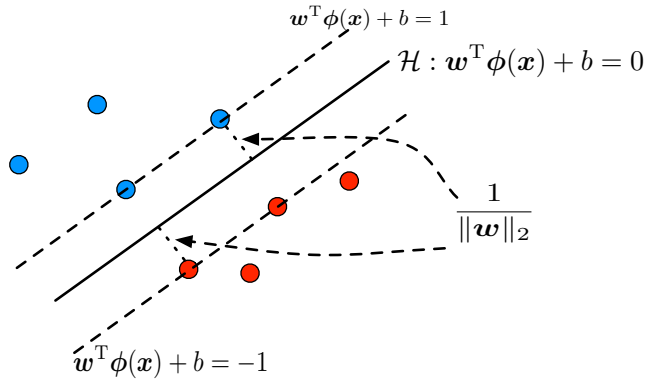
We can further constrain the problem by scaling $(\boldsymbol{w}, b)$ such that

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

We've fixed the numerator in the $\text{MARGIN}(\boldsymbol{w}, b)$ equation, and we have:

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

Hence the points closest to the decision boundary are at distance $\frac{1}{\|\boldsymbol{w}\|_2}$



SVM: max margin formulation for separable data

Assuming separable training data, we thus want to solve:

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

This is equivalent to

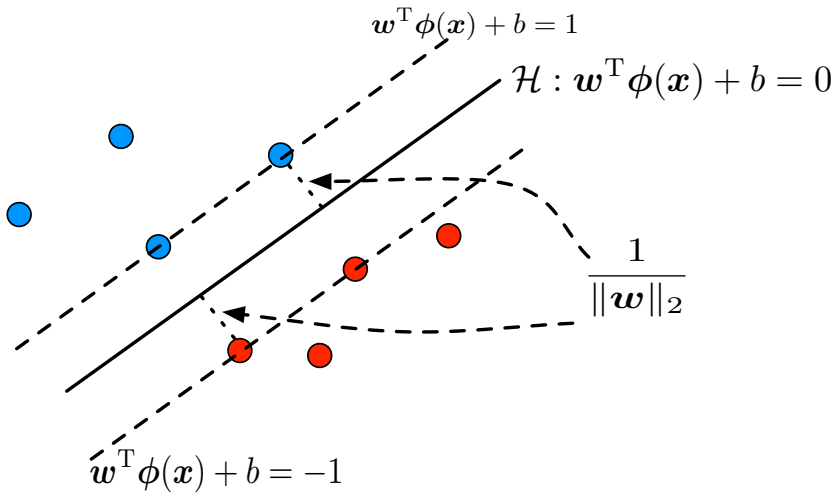
$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1, \quad \forall \quad n \end{aligned}$$

Given our geometric intuition, SVM is called a **max margin** (or large margin) classifier. The constraints are called **large margin constraints**.

Support vectors – a first look

SVM formulation for separable data

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



- "$=$": $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$, these training data are "support vectors"
- "$>$": $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] > 1$, removing them do not affect the optimal solution.

SVM for non-separable data

SVM formulation for separable data

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

Non-separable setting

In practice our training data will not be separable. What issues arise with the optimization problem above when data is not separable?

- For every $\boldsymbol{w}$ there exists a training point $\boldsymbol{x}_i$ such that

$$y_i[\boldsymbol{w}^\top \boldsymbol{x}_i + b] \leq 0$$

- There is no feasible $(\boldsymbol{w}, b)$ as at least one of our constraints is violated!

SVM for non-separable data

Constraints in separable setting

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

Constraints in non-separable setting

Idea: modify our constraints to account for non-separability! Specifically, we introduce *slack variables* $\xi_n \geq 0$:

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

- For "hard" training points, we can increase ξ_n until the above inequalities are met
- What does it mean when ξ_n is very large?

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:

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

What is the role of C?

- User-defined hyperparameter
- Trades off between the two terms in our objective
- Same idea as the regularization term in ridge regression, i.e., $C = \frac{1}{\lambda}$

How to solve this problem?

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

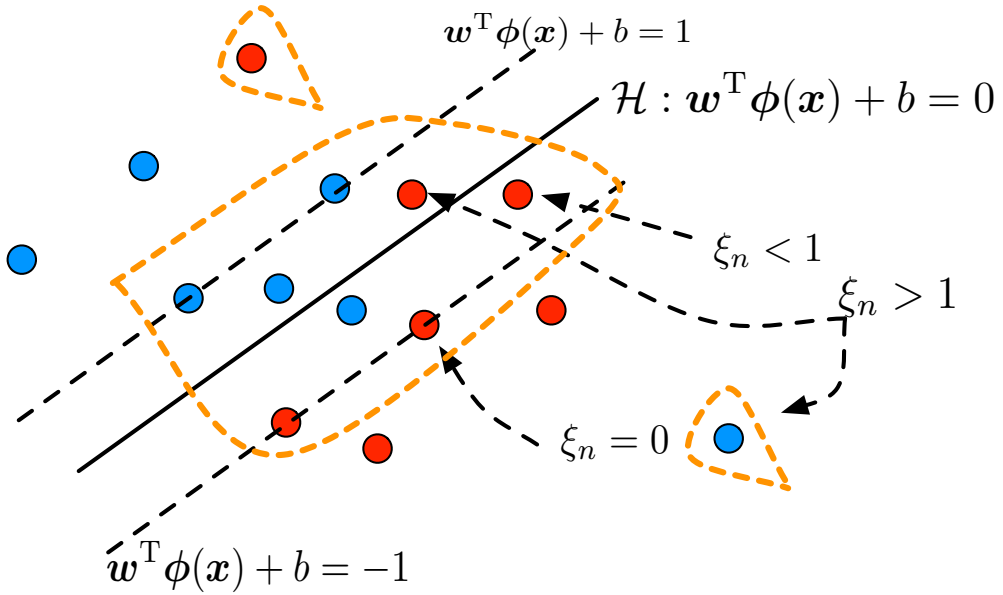
- This is a *convex quadratic program*: the objective function is quadratic in $\boldsymbol{w}$ and linear in $\boldsymbol{\xi}$ and the constraints are linear (inequality) constraints in $\boldsymbol{w}$, b and ξ_n.
- We can solve the optimization problem using general-purpose solvers, e.g., Matlab's `quadprog()` function.

Meaning of "support vectors" in SVMs

- The SVM solution is only determined by a subset of the training samples (as we will see in more detail in the next lecture)
- These samples are called support vectors
- All other training points do not affect the optimal solution, i.e., if we remove the other points and construct another SVM classifier on the reduced dataset, the optimal solution will be the same

These properties allow us to be more efficient than logistic regression or naive Bayes.

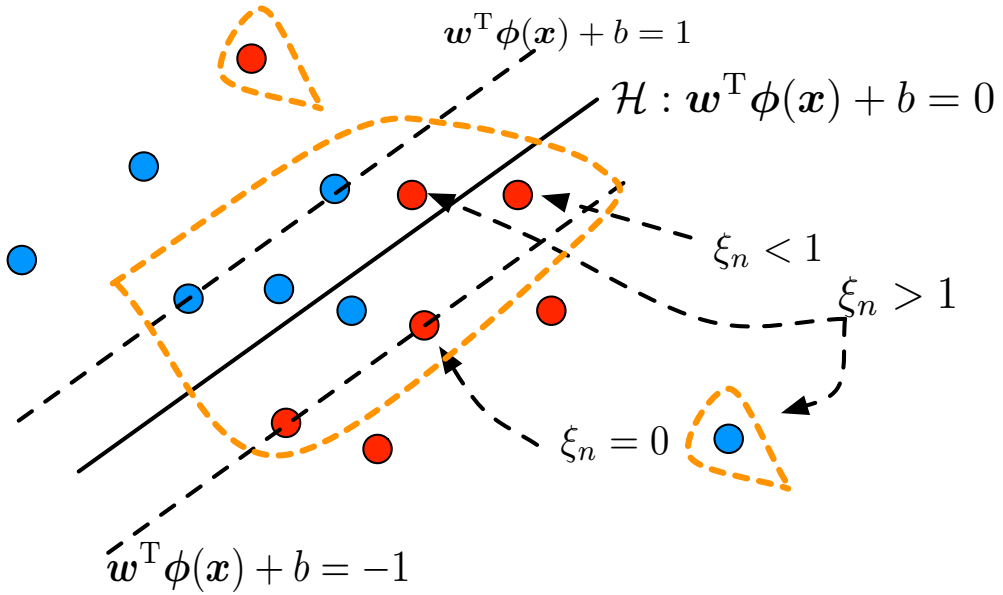
Visualization of how training data points are categorized



Support vectors are highlighted by the dotted orange lines

Recall the constraints $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n$.

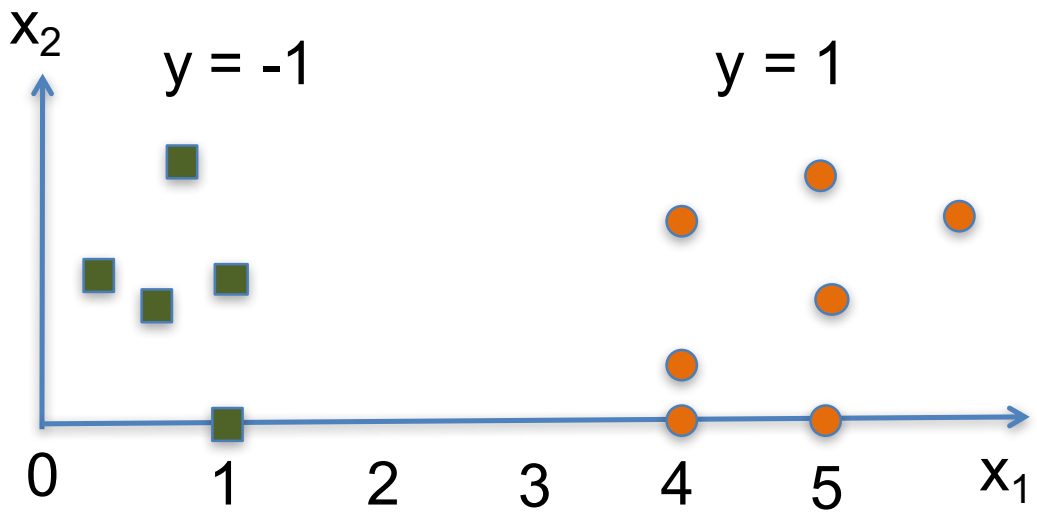
Visualization of how training data points are categorized



Recall the constraints $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n$. Three types of support vectors

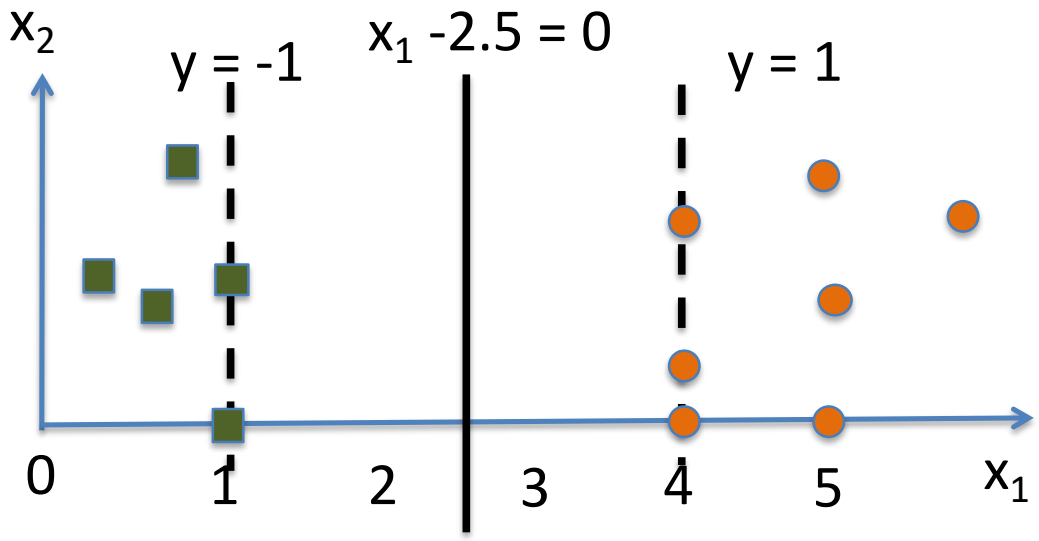
- $\xi_n = 0$: The point is on the boundary
- $0 < \xi_n \leq 1$: On the correct side, but inside the margin
- $\xi_n > 1$: On the wrong side of the boundary

Example of SVM



What will be the decision boundary learnt by solving the SVM optimization problem?

Example of SVM



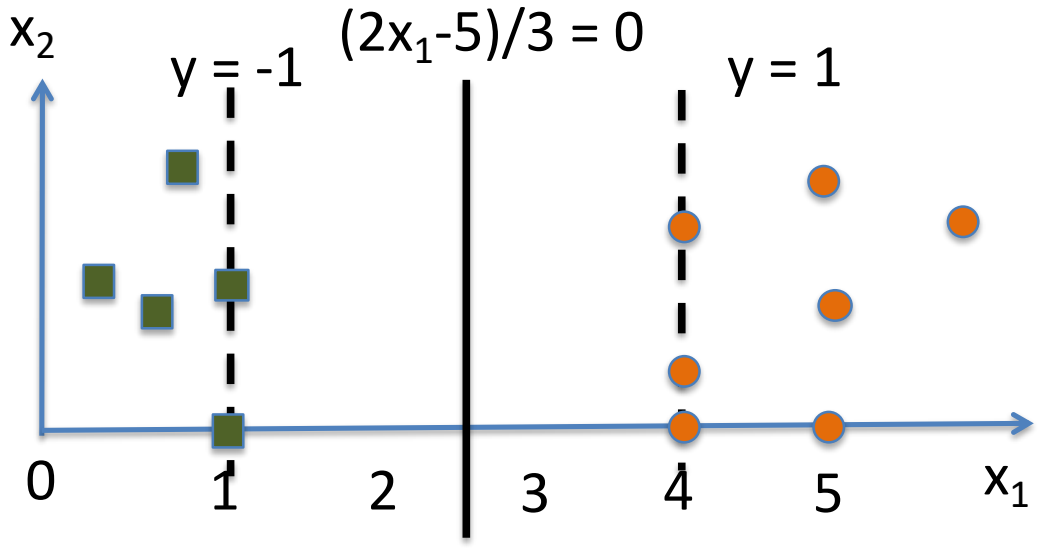
Margin $= 1.5$; the decision boundary has $\mathbf{w} = [1, 0]^\top$, and $b = -2.5$.

Is this the right scaling of $\mathbf{w}$ and b? We need the support vectors to satisfy to $y_n(\mathbf{w}^\top \mathbf{x}_n + b) = 1$.

Not quite. For example, for $\mathbf{x}_n = [1, 0]^\top$, we have

$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) = (-1)[1 - 2.5] = 1.5.$$

Example of SVM: scaling



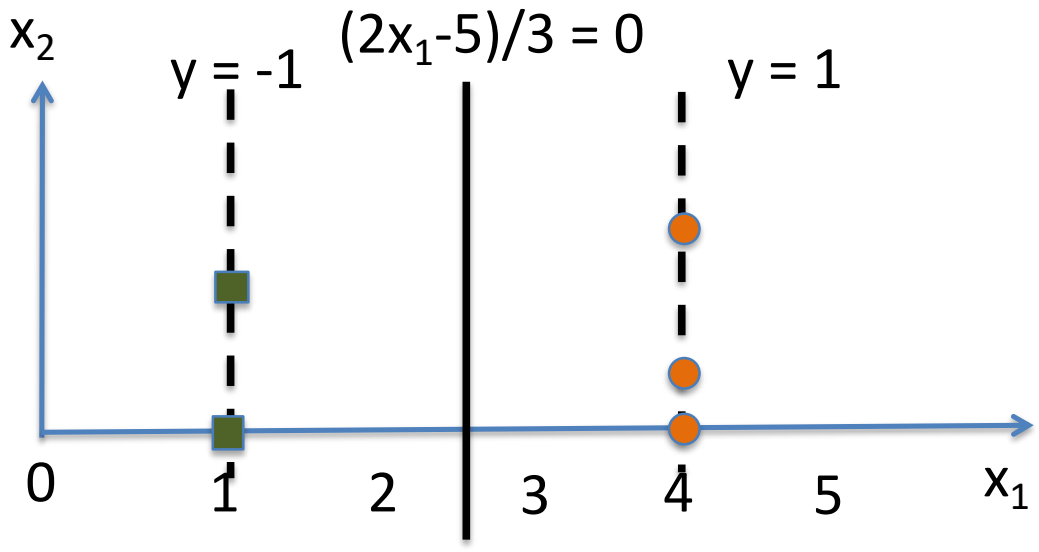
Thus, our optimization problem will re-scale $\mathbf{w}$ and b to get this equation for the same decision boundary

Margin $= 1.5$; the decision boundary has $\mathbf{w} = [2/3, 0]^\top$, and $b = -5/3$.

For example, for $\mathbf{x}_n = [1, 0]^\top$, we have

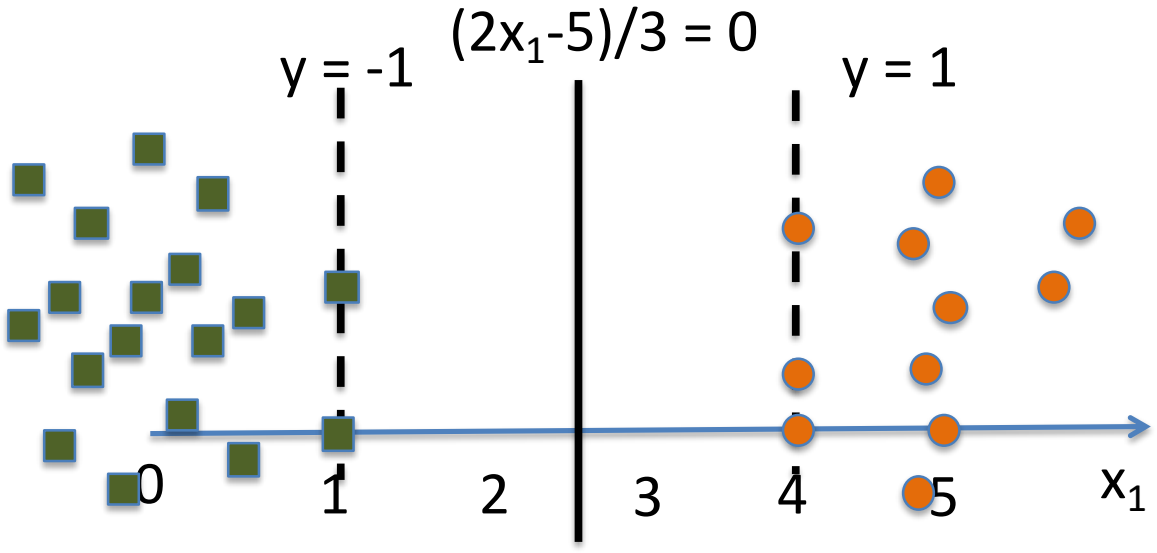
$$y_n(\mathbf{w}^\top \mathbf{x}_n + b) = (-1)[2/3 - 5/3] = 1.$$

Example of SVM: support vectors



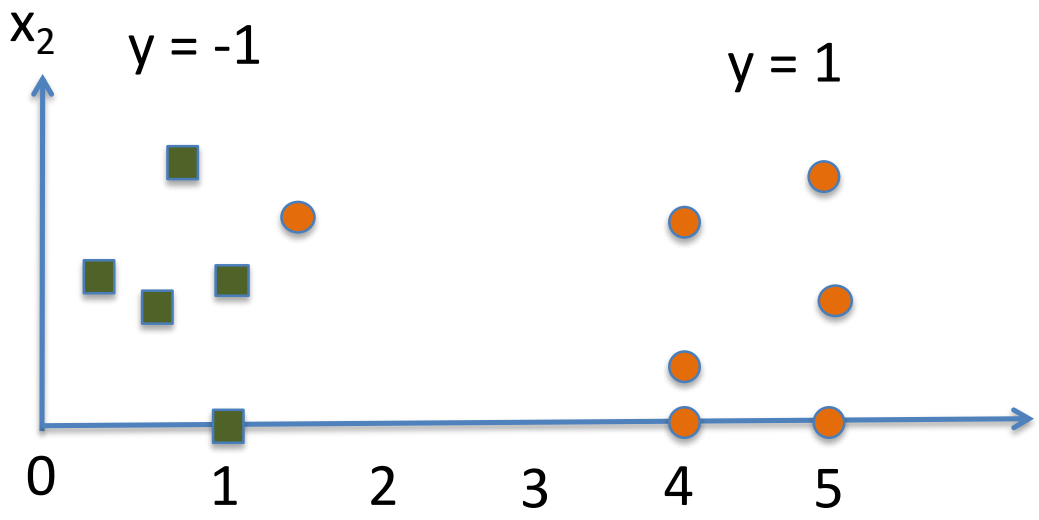
The solution to our optimization problem will be the **same** to the *reduced* dataset containing all the support vectors.

Example of SVM: support vectors



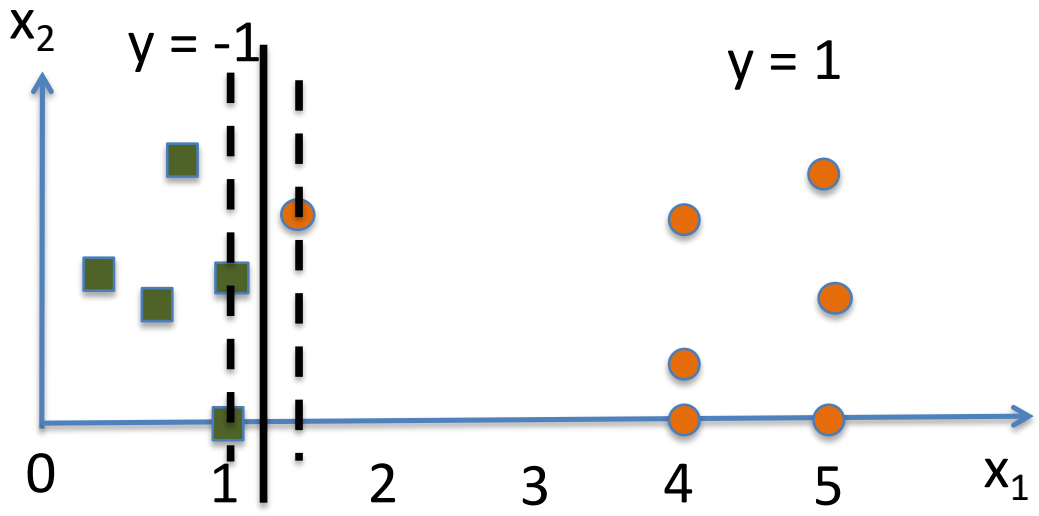
There can be many more data than the number of support vectors (so we can train on a smaller dataset).

Example of SVM: resilience to outliers



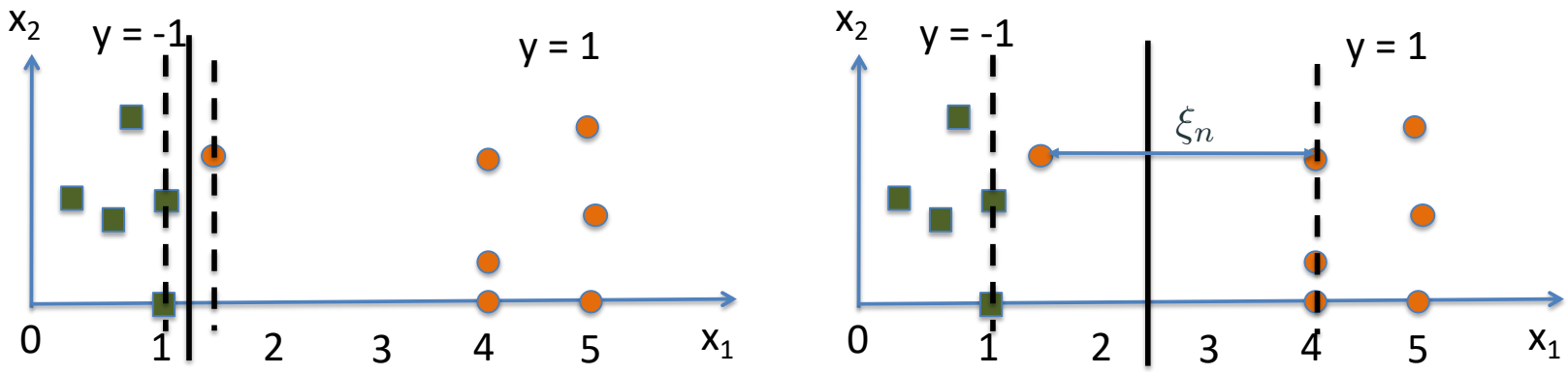
- Still linearly separable, but one of the orange dots is an "outlier".

Example of SVM: resilience to outliers



- Naively applying the hard-margin SVM will result in a classifier with small margin.
- So, better to use the soft-margin formulation.

Example of SVM: resilience to outliers



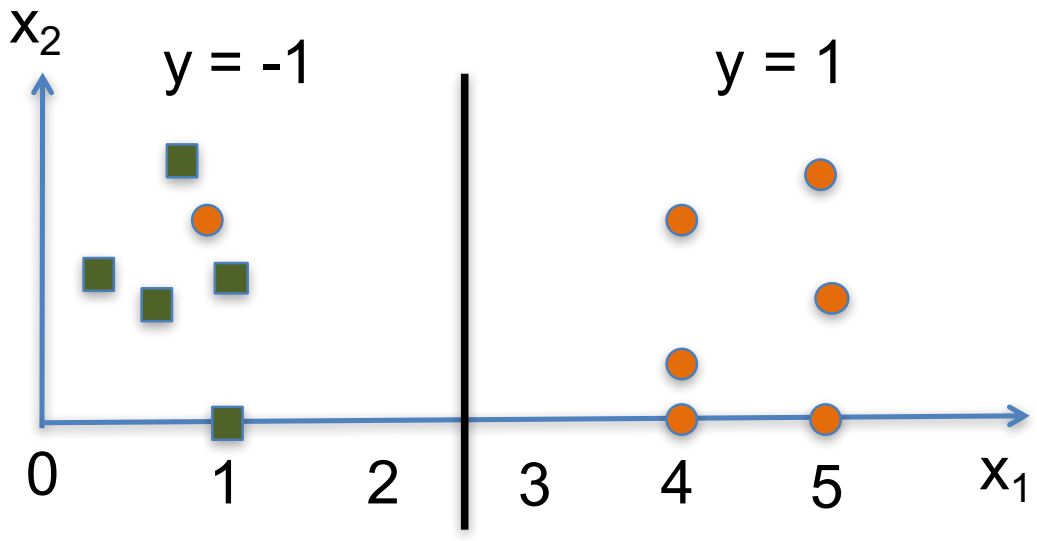
$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

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

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

- $C = \infty$ corresponds to the hard-margin SVM;
- Due to the flexibility in C, SVM is also less sensitive to outliers.

Example of SVM



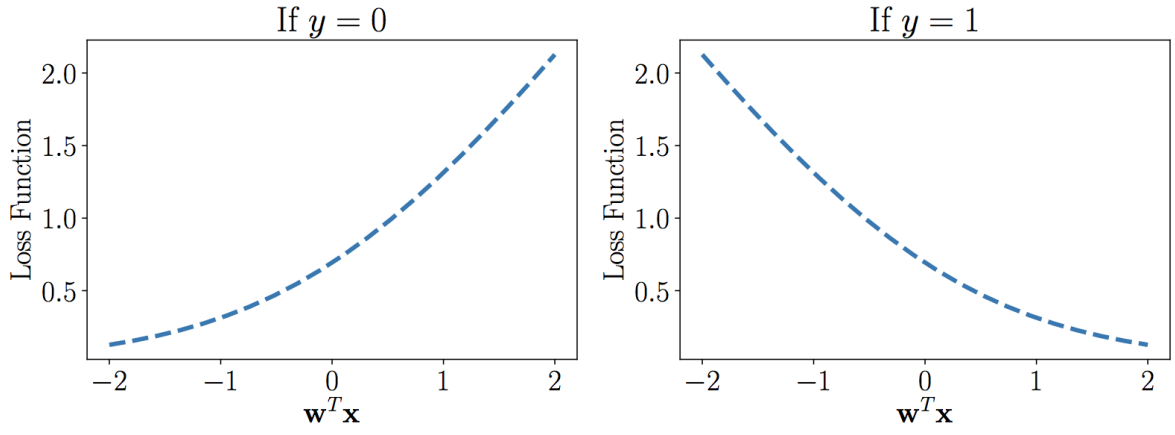
- Similar reasons apply to the case when the data is not linearly separable.
- The value of C determines how much the boundary will shift: trade-off of accuracy and robustness (sensitivity to outliers).

SVM: Hinge Loss Formulation

Logistic Regression Loss: Illustration

$$\mathcal{L}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

$$-\log\left(\frac{e^{-\mathbf{w}^T\mathbf{x_n}}}{1+e^{-\mathbf{w}^T\mathbf{x_n}}}\right) \qquad -\log\left(\frac{1}{1+e^{-\mathbf{w}^T\mathbf{x_n}}}\right)$$

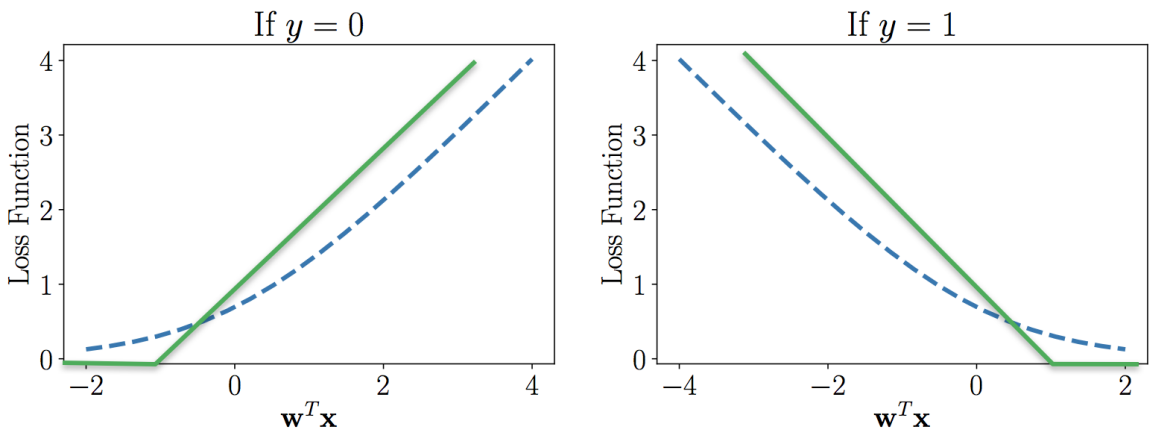


- Loss grows approx. linearly as we move away from the boundary
- Alternative: Hinge Loss Function

Hinge Loss: Illustration

$$\mathcal{L}(\boldsymbol{w}) = -\sum_n \{y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log[1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]\}$$

$\max\left(\mathbf{w}^T\mathbf{x_n} + 1, 0\right)$ $\max\left(1 - \mathbf{w}^T\mathbf{x_n}, 0\right)$

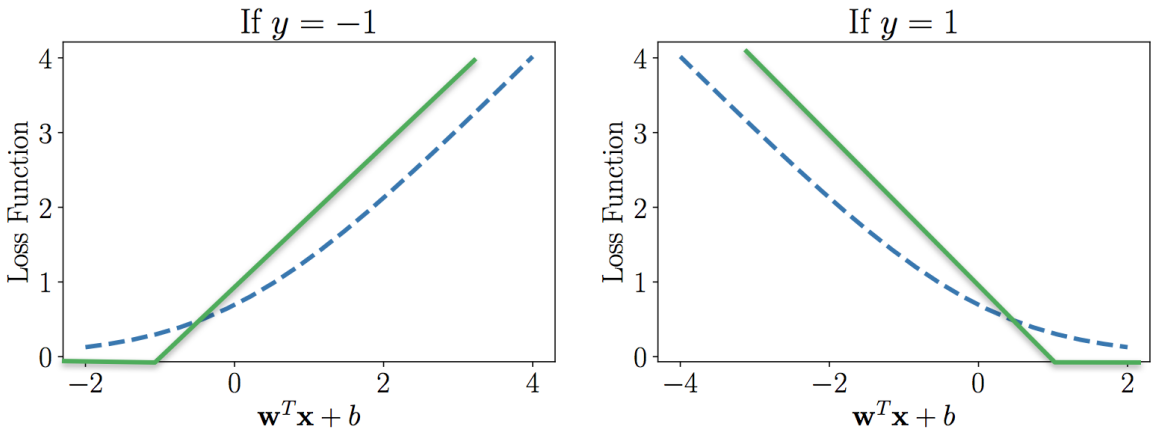


- Loss grows linearly as we move away from the boundary
- No penalty if a point is more than 1 unit from the boundary
- Makes the search for the boundary easier (as we will see later)

Hinge Loss: Mathematical Expression

$$\mathcal{L}(\boldsymbol{w}) = -\sum_{n} \max(0, 1 - y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b))$$

$\max\left(\mathbf{w}^T\mathbf{x_n} + b + 1, 0\right)$ $\max\left(1 - \mathbf{w}^T\mathbf{x_n} - b, 0\right)$



- Change of notation $y = 0 \rightarrow y = -1$
- Separate the bias term b from $\mathbf{w}$
- Makes the mathematical expression more compact

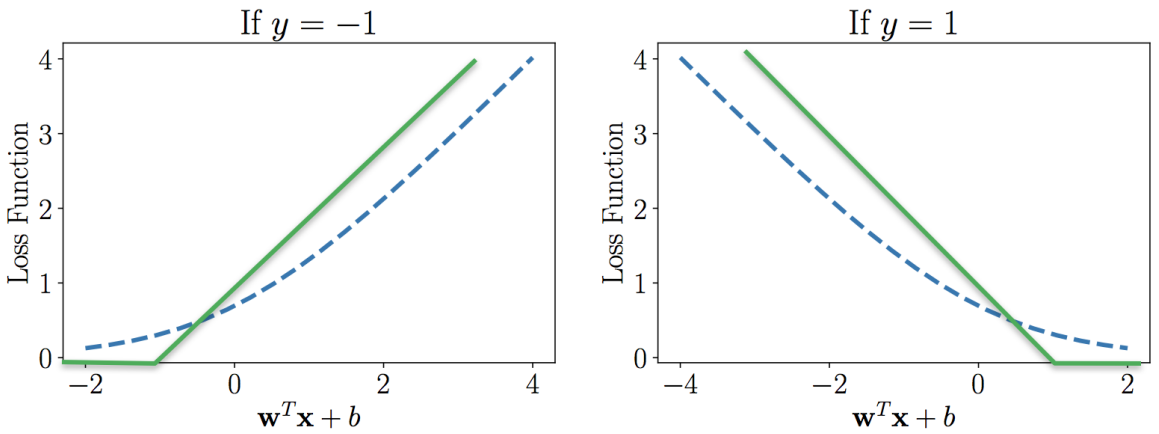
Hinge Loss: Mathematical Expression

Definition

Assume $y \in \{-1, 1\}$ and the decision rule is $h(\boldsymbol{x}) = \text{SIGN}(\boldsymbol{w}^\top \boldsymbol{x})$ with $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$,

$$\ell^{\text{HINGE}}(f(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } yf(\boldsymbol{x}) \geq 1 \\ 1 - yf(\boldsymbol{x}) & \text{otherwise} \end{cases}$$

$\max\left(\mathbf{w}^T\mathbf{x_n} + b + 1, 0\right)$ $\max\left(1 - \mathbf{w}^T\mathbf{x_n} - b, 0\right)$



Hinge loss

Definition

Assume $y \in \{-1, 1\}$ and the decision rule is $h(\boldsymbol{x}) = \text{SIGN}(f(\boldsymbol{x}))$ with $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$,

$$\ell^{\text{HINGE}}(f(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } yf(\boldsymbol{x}) \geq 1 \\ 1 - yf(\boldsymbol{x}) & \text{otherwise} \end{cases}$$

Intuition

- No penalty if raw output, $f(\boldsymbol{x})$, has same sign and is far enough from decision boundary (i.e., if 'margin' is large enough)
- Otherwise pay a growing penalty, between 0 and 1 if signs match, and greater than one otherwise

Convenient shorthand

$$\ell^{\text{HINGE}}(f(\boldsymbol{x}), y) = \max(0, 1 - yf(\boldsymbol{x})) = (1 - yf(\boldsymbol{x}))_+$$

Optimization Problem of support vector machines (SVM)

Minimizing the total hinge loss on all the training data

$$\min_{\boldsymbol{w},b} \sum_n \underbrace{\max(0, 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b])}_{\text{hinge loss for sample } n} + \underbrace{\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2}_{\text{regularizer}}$$

Analogous to regularized least squares, as we balance between two terms (the loss and the regularizer).

- Can solve using gradient descent to get the optimal $\mathbf{w}$ and b
- Gradient of the first term will be either 0, $\mathbf{x}_n$ or $-\mathbf{x}_n$ depending on y_n and $\boldsymbol{w}^\top \boldsymbol{x}_n + b$
- Much easier to compute than in logistic regression, where we need to compute the sigmoid function $\sigma(\boldsymbol{w}^\top \boldsymbol{x}_n + b)$ in each iteration

Equivalence of These Two Formulations

Recovering our previous SVM formulation

Rewrite the geometric formulation as the hinge loss formulation:

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

Here's the geometric formulation again:

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

Now since $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n \iff \xi_n \geq 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]$:

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

Now since the ξ_n should always be as small as possible, we obtain:

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

Advantages of SVM

We've seen that the geometric formulation of SVM is equivalent to minimizing the empirical hinge loss. This explains why SVM:

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

We will need to use duality to show the next three properties.

Lagrange Duality and KKT conditions

What is duality?

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.

- Dual problem is always convex–easy to solve.
- Primal and dual problems are not always equivalent.
- Gives a systematic way of solving for the optimal variables.
- Dual variables tell us "how bad" constraints are.

The following material requires some advanced concepts. The main point you should understand is that we will solve the dual SVM problem in lieu of the max margin (primal) formulation

Constrained Optimization

$$\begin{cases} \min_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{s.t.} & g_i(\boldsymbol{x}) \leq 0, \quad \forall\ i \\ & h_i(\boldsymbol{x}) = 0, \quad \forall\ j \end{cases}$$

This is the 'primal' problem. The generalized Lagrangian is defined as follows:

$$L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\boldsymbol{x}) + \sum_i \alpha_i g_i(\boldsymbol{x}) + \sum_j \beta_j h_j(\boldsymbol{x})$$

Consider the following function:

$$\theta_P(\boldsymbol{x}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}, \alpha_i \geq 0} L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

- If $\boldsymbol{x}$ violates a primal constraint, $\theta_P(\boldsymbol{x}) = \infty$; otherwise $\theta_P(\boldsymbol{x}) = f(\boldsymbol{x})$
- Thus $\min_{\boldsymbol{x}} \theta_P(\boldsymbol{x}) = \min_{\boldsymbol{x}} \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}, \alpha_i \geq 0} L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ has same solution as the primal problem, which we denote as p^*

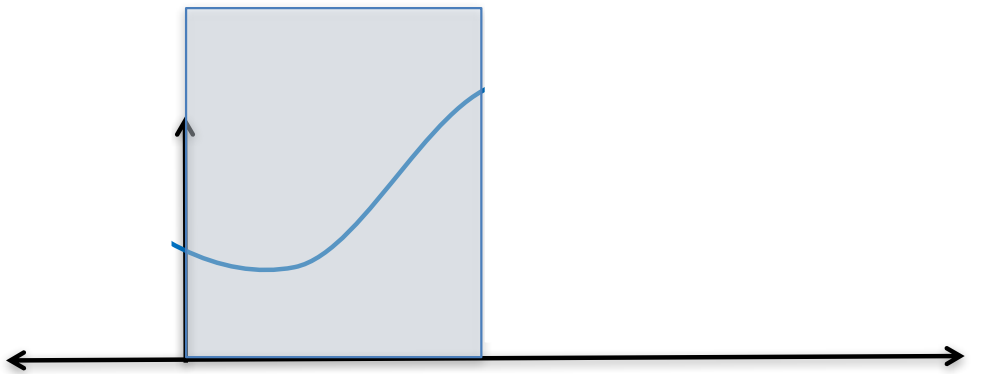
Constrained Optimization

$$\begin{cases} \min_{\boldsymbol{x}} & f(\boldsymbol{x}) \\ \text{s.t.} & g_i(\boldsymbol{x}) \leq 0, \quad \forall\ i \\ & h_i(\boldsymbol{x}) = 0, \quad \forall\ j \end{cases}$$

This is the 'primal' problem.

$$\theta_P(\boldsymbol{x}) = \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0} L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$\max_{\alpha>0,\beta} L(x, \alpha, \beta)$ is equal to f(x) for the feasible x and infinity everywhere else



Constrained Optimization – Inequality Constraints

Primal Problem

$$p^* = \min_{\boldsymbol{x}} \theta_P(\boldsymbol{x}) = \min_{\boldsymbol{x}} \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0} L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

Dual Problem

Consider the function:

$$\theta_D(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

$$d^* = \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0} \theta_D(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \max_{\boldsymbol{\alpha},\boldsymbol{\beta},\alpha_i \geq 0} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta})$$

Relationship between primal and dual?

- $p^* \geq d^*$ (weak duality)
- 'min max' of any function is always greater than the 'max min'
- `https://en.wikipedia.org/wiki/Max%E2%80%93min_inequality`

Strong Duality

When $p^* = d^*$, we can solve the dual problem in lieu of primal problem!

Sufficient conditions for strong duality:

- f and g_i are convex, h_i are affine (i.e., linear with offset)
- Inequality constraints are strictly 'feasible,' i.e., there exists some $\boldsymbol{x}$ such that $g_i(\boldsymbol{x}) < 0$ for all i
- These conditions are all satisfied by the SVM optimization problem!

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

Strictly feasible solution if ξ_n are all sufficiently large.

How to find the solution $p^* = d^*$? Use KKT Conditions

Strong duality implies that there exist $\boldsymbol{x}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$ such that:

- $\boldsymbol{x}^*$ is the solution to the primal and $\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$ is the solution to the dual
- $p^* = d^* = L(\boldsymbol{x}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$
- $\boldsymbol{x}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$ satisfy the KKT conditions (which in fact are necessary and sufficient for optimality)

The Karush-Kuhn-Tucker (KKT) conditions are:

- Stationarity: $\frac{\partial L(\boldsymbol{x}, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)}{\partial \boldsymbol{x}}|_{\boldsymbol{x}^*} = 0$. $\boldsymbol{x}^*$ is a local extremum of the Lagrangian L for fixed $\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$.
- Feasibility: $g_i(\boldsymbol{x}^*) \leq 0$ and $h_i(\boldsymbol{x}^*) = 0$ (primal) and $\alpha_i^* \geq 0$ (dual) for all i. All primal and dual constraints are satisfied.
- Complementary slackness: $\alpha_i^* g_i(\boldsymbol{x}^*) = 0$ for all i. Either the Lagrange multiplier α_i^* is 0, or the corresponding constraint $g_i(\boldsymbol{x}^*) \leq 0$ is tight (i.e., $g_i(\boldsymbol{x}^*) = 0$).

A simple example

Consider the example of convex quadratic programming

$$\begin{aligned} \min \quad & \frac{1}{2}x^2 \\ \text{s.t.} \quad & -x \leq 0 \\ & 2x - 3 \leq 0 \end{aligned}$$

The generalized Lagrangian is (note that we do not have equality constraints)

$$L(x, \alpha) = \frac{1}{2}x^2 + \alpha_1(-x) + \alpha_2(2x - 3) = \frac{1}{2}x^2 + (2\alpha_2 - \alpha_1)x - 3\alpha_2$$

under the constraints that $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$. Its dual problem is

$$\max_{\alpha_1 \geq 0, \alpha_2 \geq 0} \min_x L(x, \alpha) = \max_{\alpha_1 \geq 0, \alpha_2 \geq 0} \min_x \frac{1}{2}x^2 + (2\alpha_2 - \alpha_1)x - 3\alpha_2$$

Example (cont'd)

We now solve $\min_x L(x, \alpha)$. The optimal x is attained by

$$\frac{\partial(\frac{1}{2}x^2 + (2\alpha_2 - \alpha_1)x - 3\alpha_2)}{\partial x} = 0 \rightarrow x = -(2\alpha_2 - \alpha_1)$$

We next substitute the solution back into the Lagrangian:

$$g(\alpha) = \min_x \frac{1}{2}x^2 + (2\alpha_2 - \alpha_1)x - 3\alpha_2 = -\frac{1}{2}(2\alpha_2 - \alpha_1)^2 - 3\alpha_2$$

Our dual problem can now be simplified:

$$\max_{\alpha_1 \geq 0, \alpha_2 \geq 0} -\frac{1}{2}(2\alpha_2 - \alpha_1)^2 - 3\alpha_2$$

We will solve the dual next.

Solving the dual

Note that,

$$g(\alpha) = -\frac{1}{2}(2\alpha_2 - \alpha_1)^2 - 3\alpha_2 \leq 0$$

for all $\alpha_1 \geq 0, \alpha_2 \geq 0$. Thus, to maximize the function, the optimal solution is

$$\alpha_1^* = 0, \quad \alpha_2^* = 0$$

This brings us back the optimal solution of x

$$x^* = -(2\alpha_2^* - \alpha_1^*) = 0$$

Namely, we have arrived at the same solution as the one we guessed from the primal formulation

The Diet problem

Nutrients	Food 1 ($2)	Food 2 ($5)	Food 3 ($15)
Carbs	20	1	1
Protein	1	30	40
Vitamins	1	10	5

To satisfy daily nutritional requirements we need at least

- 200 units of carbs
- 50 units of protein
- 40 units of vitamins

Primal problem: How do we minimize the cost of satisfying these requirements?

The Diet problem

Nutrients	Food 1 ($2)	Food 2 ($5)	Food 3 ($15)
Carbs	20	1	1
Protein	1	30	40
Vitamins	1	10	5

$$\begin{aligned} \min_{x_1,x_2,x_3} \quad & 2x_1 + 5x_2 + 15x_3 \\ \text{s.t.} \quad & -20x_1 - x_2 - x_3 \leq -200 \\ & -x_1 - 30x_2 - 40x_3 \leq -50 \\ & -x_1 - 10x_2 - 5x_3 \leq -40 \end{aligned}$$

Primal Solution: $x_1 \approx 9.84$, $x_2 \approx 3$, $x_3 = 0$
Dual Solution: $\alpha_1 = 0.07$, $\alpha_2 = 0$, $\alpha_3 = 0.5$

$\alpha_2 = 0$ means that protein requirement is easy to satisfy

The Diet problem: Dual

Nutrients	Food 1 ($2)	Food 2 ($5)	Food 3 ($15)
Carbs	20	1	1
Protein	1	30	40
Vitamins	1	10	5

A pharmacist wants to create a diet pill to satisfy the requirements and maximize profit. α_1, α_2, α_3 are the shadow prices of the nutrients.

$$\begin{aligned} \max_{\alpha_1,\alpha_2,\alpha_3} \quad & 200\alpha_1 + 50\alpha_2 + 40\alpha_3 \\ \text{s.t.} \quad & 20\alpha_1 + \alpha_2 + \alpha_3 \leq 2 \\ & \alpha_1 + 30\alpha_2 + 10\alpha_3 \leq 5 \\ & \alpha_1 + 40\alpha_2 + 5\alpha_3 \leq 15 \\ & \alpha_1, \alpha_2, \alpha_3 \geq 0 \end{aligned}$$

Primal Solution: $x_1 \approx 9.84$, $x_2 \approx 3$, $x_3 = 0$
Dual Solution: $\alpha_1 = 0.07$, $\alpha_2 = 0$, $\alpha_3 = 0.5$

Recap

- When working with constrained optimization problems with inequality constraints, we can write down primal and dual problems.
- The dual solution is always a lower bound on the primal solution (weak duality) and it is always convex
- The duality gap equals 0 under certain conditions (strong duality), and in such cases we can either solve the primal or dual problem.
- Strong duality (and thus the KKT conditions) hold for the SVM problem.
- See `http://cs229.stanford.edu/notes/cs229-notes3.pdf` for details

Dual Derivation of SVMs

Derivation of the dual

We will next derive the dual formulation for SVMs.

Recipe

1. Formulate the generalized Lagrangian function that incorporates the constraints and introduces dual variables
2. Minimize the Lagrangian function over the primal variables
3. Substitute the primal variables for dual variables in the Lagrangian
4. Maximize the Lagrangian with respect to dual variables
5. Recover the solution (for the primal variables) from the dual variables

Deriving the dual for SVM

Primal SVM

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

The constraints are equivalent to $-\xi_n \leq 0$ and $1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] - \xi_n \leq 0$.

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 $\lambda_n \geq 0$.

Minimizing the Lagrangian

Taking derivatives with respect to the primal variables

$$\frac{\partial L}{\partial \boldsymbol{w}} = \frac{\partial}{\partial \boldsymbol{w}} \left(\frac{1}{2} \|\boldsymbol{w}\|_2^2 - \sum_n \alpha_n y_n \boldsymbol{w}^\top \boldsymbol{x}_n \right) = \boldsymbol{w} - \sum_n y_n \alpha_n \boldsymbol{x}_n = 0$$

$$\frac{\partial L}{\partial b} = \frac{\partial}{\partial b} - \sum_n \alpha_n y_n b = - \sum_n \alpha_n y_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = \frac{\partial}{\partial \xi_n} (C - \lambda_n - \alpha_n) \xi_n = C - \lambda_n - \alpha_n = 0$$

These equations link the primal variables and the dual variables and provide new constraints on the dual variables:

$$\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{x}_n$$

$$\sum_n \alpha_n y_n = 0$$

$$C - \lambda_n - \alpha_n = 0$$

Rearrange the Lagrangian

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

where $\alpha_n \geq 0$ and $\lambda_n \geq 0$. We now know that $\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{x}_n$.

$$\begin{aligned} g(\{\alpha_n\},\{\lambda_n\}) &= L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) \\ &= \underbrace{\sum_n (C - \alpha_n - \lambda_n)\xi_n}_{\text{gather terms with } \xi_n} + \frac{1}{2}\| \underbrace{\sum_n y_n \alpha_n \boldsymbol{x}_n}_{\text{substitute for } \boldsymbol{w}} \|_2^2 + \sum_n \alpha_n \\ &\quad - \sum_n \alpha_n y_n \underbrace{\left(\sum_m y_m \alpha_m \boldsymbol{x}_m\right)^\top}_{\text{again substitute for } \boldsymbol{w}} \boldsymbol{x}_n - \left(\sum_n \alpha_n y_n\right) b \end{aligned}$$

Then, set $\sum_n \alpha_n y_n = 0$ and $C - \lambda_n - \alpha_n = 0$ and simplify to get..

The dual problem

Maximizing the dual under the constraints

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

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

$$\sum_n \alpha_n y_n = 0$$

$$C - \lambda_n - \alpha_n = 0, \quad \forall\ n$$

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

We can simplify as the objective function does not depend on λ_n. Specifically, we can combine the constraints involving λ_n resulting in the following inequality constraint: $\alpha_n \leq C$:

$$C - \lambda_n - \alpha_n = 0,\ \lambda_n \geq 0 \iff \lambda_n = C - \alpha_n \geq 0$$

$$\iff \alpha_n \leq C$$

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
- Independent of the size d of $\mathbf{x}$: SVM scales better for high-dimensional feature.
- May seem like a lot of optimization variables when N is large, but many of the α_n's become zero. α_n is non-zero only if the n^{th} point is a support vector

Why do many α_n's become zero?

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

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

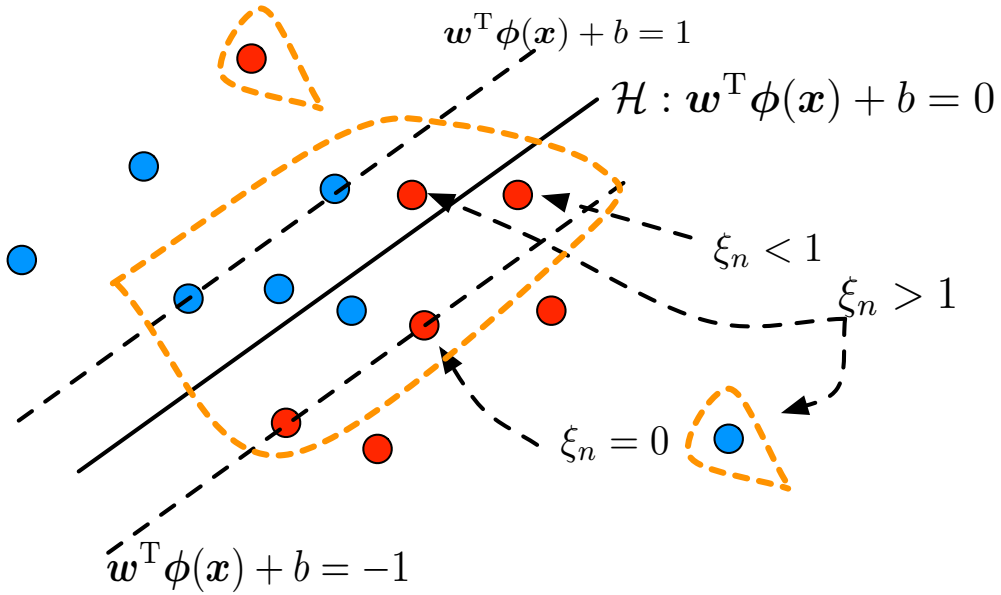
$$\sum_n \alpha_n y_n = 0$$

- KKT complementary slackness conditions tell us:

$$(1)\ \lambda_n \xi_n = 0 \qquad (2)\ \alpha_n \{1 - \xi_n - y_n [\boldsymbol{w}^\top \boldsymbol{x}_n + b]\} = 0$$

- (2) tells us that $\alpha_n > 0$ iff $1 - \xi_n = y_n [\boldsymbol{w}^\top \boldsymbol{x}_n + b]$
  - If $\xi_n = 0$, then support vector is on the margin
  - Otherwise, $\xi_n > 0$ means that the point is an outlier

Visualizing the support vectors



Support vectors ($\alpha_n > 0$) are highlighted by the dotted orange lines.

- $\xi_n = 0$ and $0 < \alpha_n < C$ when $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] = 1$.
- $\xi_n > 0$ and $\alpha_n = C$ if $y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] < 1$.

Once we solve for α_n's, how to get w and b?

Recovering **w**

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

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

Recovering b

If $0 < \alpha_n < C$ and $y_n \in \{-1, 1\}$:

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

$$b = y_n - \boldsymbol{w}^\top \boldsymbol{x}_n$$

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

Advantages of SVM

We've seen that the geometric formulation of SVM is equivalent to minimizing the empirical hinge loss. This explains why SVM:

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

The last thing left to consider is non-linear decision boundaries, or kernel SVMs

Kernel SVM

Non-linear basis functions in SVM

- What if the data is not linearly separable?
- We can transform the feature vector $\mathbf{x}$ using non-linear basis functions. For example,

$$\phi(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

- Replace $\mathbf{x}$ by $\phi(\mathbf{x})$ in both the primal and dual SVM formulations

Primal and Dual SVM Formulations: Kernel Versions

Primal

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

Dual

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

IMPORTANT POINT: In the dual problem, we only need $\boldsymbol{\phi}(\boldsymbol{x}_m)^\top \boldsymbol{\phi}(\boldsymbol{x}_n)$.

Dual Kernel SVM

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

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

- $k(\mathbf{x}_m, \mathbf{x}_n)$ is a scalar and it is independent of the dimension of the feature vector $\phi(\mathbf{x})$.
- $k(\mathbf{x}_m, \mathbf{x}_n)$ roughly measures the similarity of $\mathbf{x}_m$ and $\mathbf{x}_n$.
- $k(\mathbf{x}_m, \mathbf{x}_n)$ is a kernel function if it is symmetric and positive-definite ($k(\mathbf{x}, \mathbf{x}) > 0$ for all $\mathbf{x} > 0$).

Examples of popular kernel functions

We do not need to know the exact form of $\phi(\boldsymbol{x})$, which lets us define much more flexible nonlinearities.

- Dot product:

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

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

$$k(\mathbf{x}_m, \mathbf{x}_n) = \mathbf{x}_m^\top \mathbf{Q} \mathbf{x}_n$$

- Polynomial kernels:

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

- Radial basis function:

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

and many more.

Test prediction

Learning w and b:

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

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

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 $\boldsymbol{w}^\top \phi(\boldsymbol{x}) + b$? Fortunately, no!

Test Prediction:

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

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

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 ?

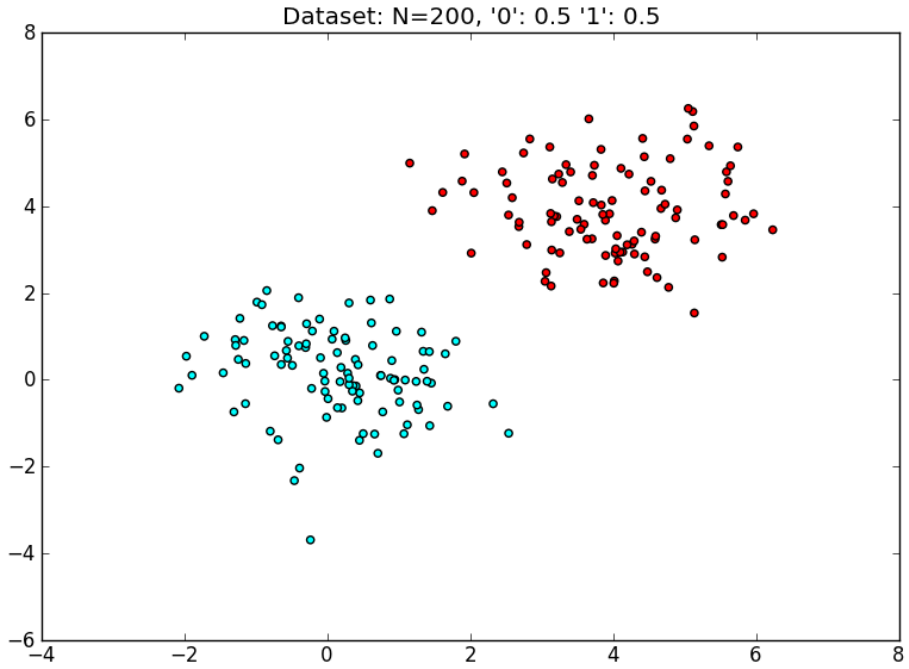


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

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

Here is the decision boundary with linear soft-margin SVM

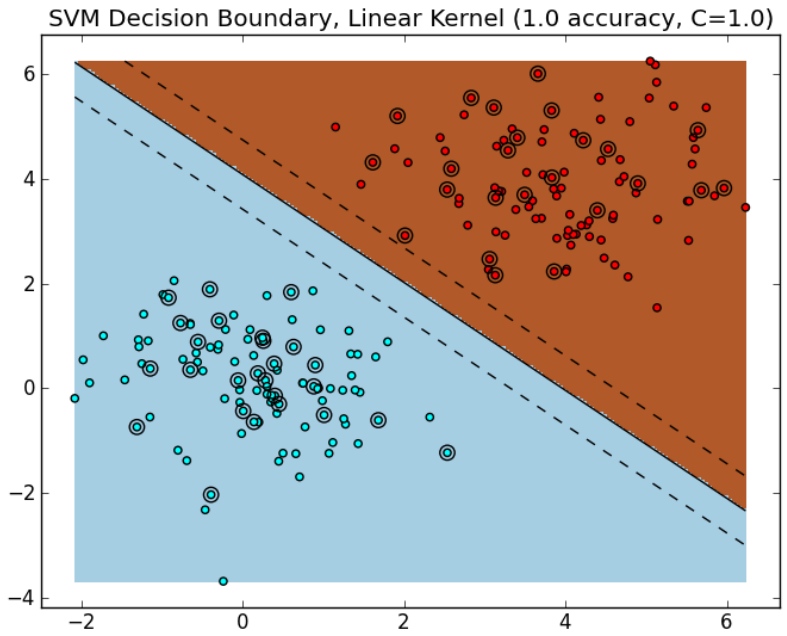


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

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?

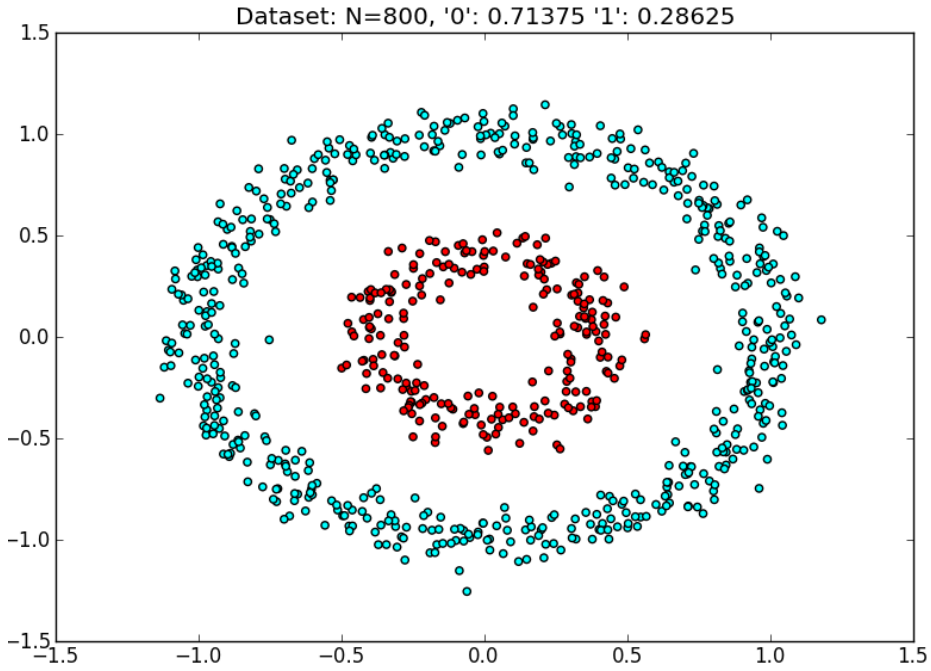


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

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

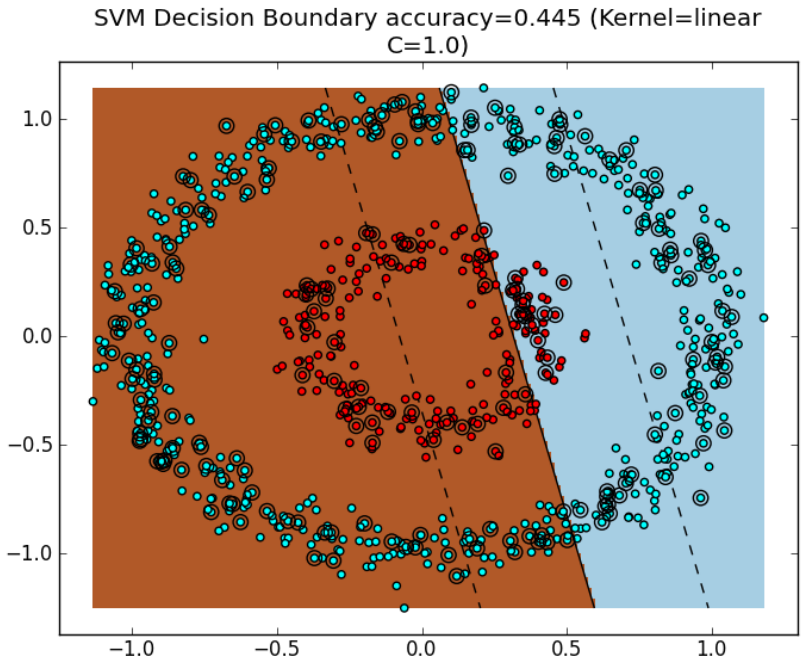
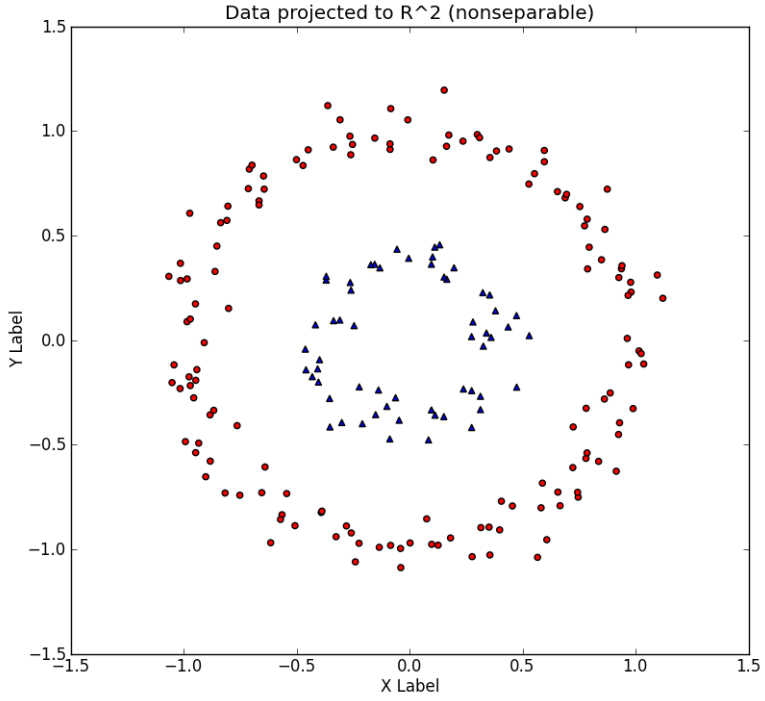


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

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



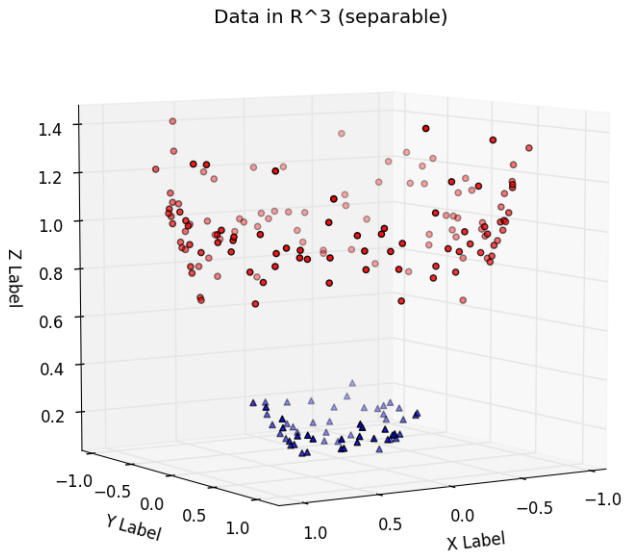


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

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

Then find the decision boundary. How? Solve the Dual problem

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

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

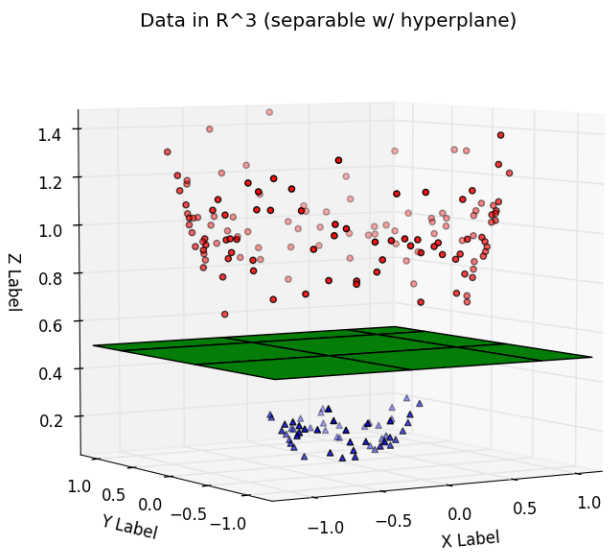
$$\sum_n \alpha_n y_n = 0$$

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

Example of Kernel SVM

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

Here is the resulting decision boundary



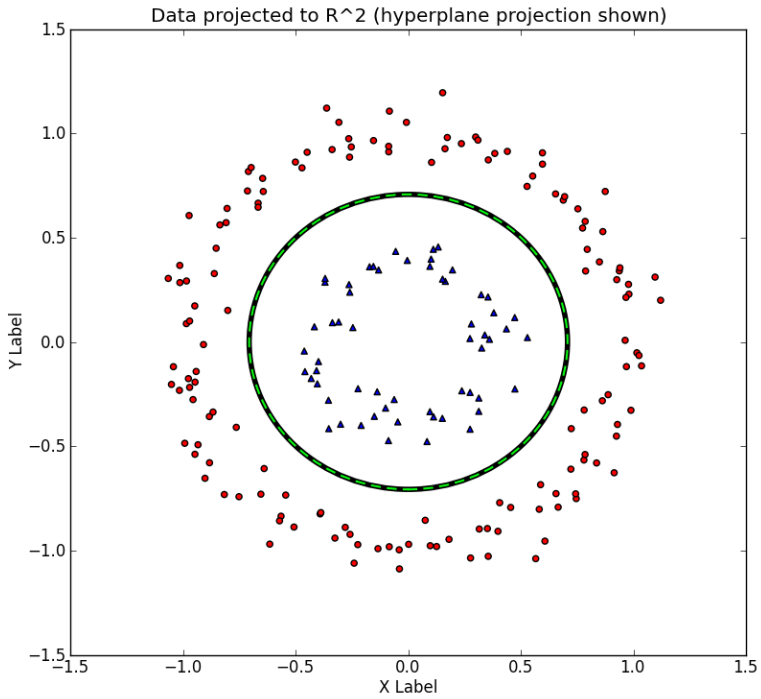


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

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

In general, you don't need to concretely define $\phi(\mathbf{x})$. In the dual problem we can just use the kernel function $k(\mathbf{x}_m, \mathbf{x}_n)$. For cases where $\phi(\mathbf{x})$ is concretely defined, $k(\mathbf{x}_m, \mathbf{x}_n) = \phi(\mathbf{x}_m)^T \phi(\mathbf{x}_n)$.

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

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

$$\sum_n \alpha_n y_n = 0$$

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: Polynomial kernel (degree 4)

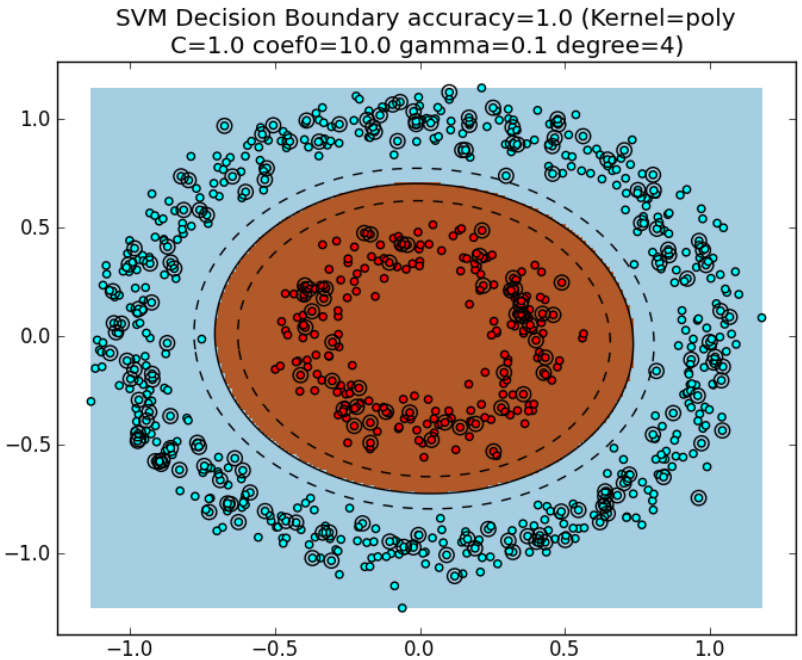


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

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

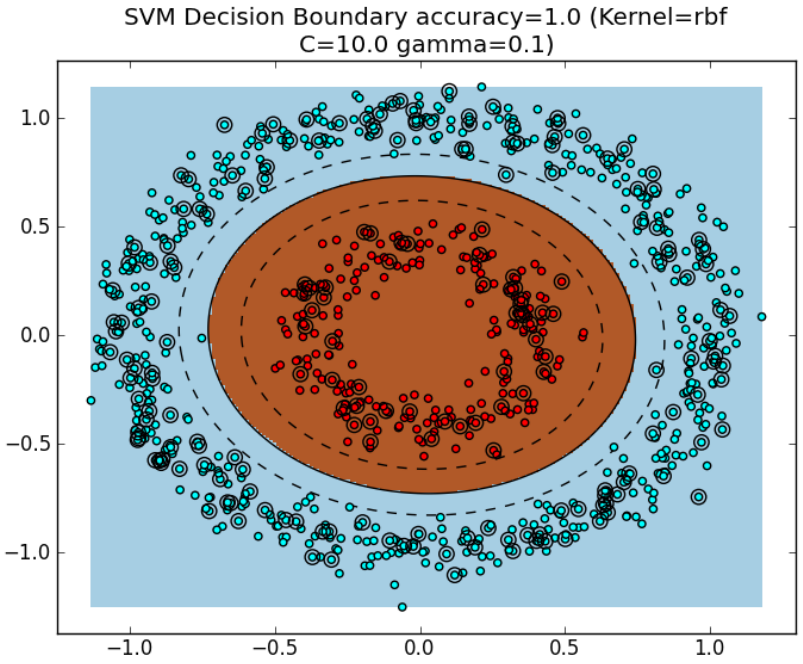


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Summary

You should know:

- Hinge loss function of SVM.
- How to derive the SVM dual.
- How to use the "kernel trick" in the dual SVM formulation to enable kernel SVM.
- How to compute an SVM prediction.