Support Vector Machines, Kernel SVM

Professor Ameet Talwalkar

Outline

1. Administration
2. Review of last lecture
3. SVM – Hinge loss (primal formulation)
4. Kernel SVM

Announcements

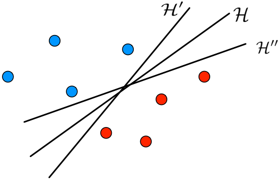
- HW4 due now
- HW5 will be posted online today
- Midterm has been graded
  - Average: 64.6/90
  - Median: 64.5/90
  - Standard Deviation: 14.8

Outline

1. Administration
2. Review of last lecture
   - SVMs – Geometric interpretation
3. SVM – Hinge loss (primal formulation)
4. Kernel SVM

SVM Intuition: where to put the decision boundary?

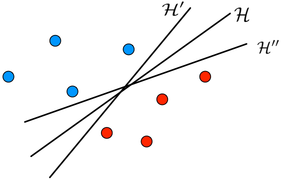
Consider the following *separable* training dataset, i.e., we assume there exists a decision boundary that separates the two classes perfectly. There are an *infinite* number of decision boundaries $\mathcal{H} : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 0$!



Which one should we pick?

SVM Intuition: where to put the decision boundary?

Consider the following *separable* training dataset, i.e., we assume there exists a decision boundary that separates the two classes perfectly. There are an *infinite* number of decision boundaries $\mathcal{H} : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b = 0$!



Which one should we pick? Idea: Find a decision boundary in the '*middle*' of the two classes. In other words, we want a decision boundary that:

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

Distance from a point to decision boundary

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

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

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

Distance from a point to decision boundary

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

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

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

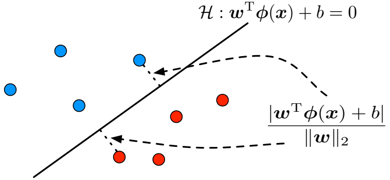
Namely, $(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b)$ and $\boldsymbol{x}$'s label y must have the same sign, so:

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

Optimizing the Margin

Margin Smallest distance between the hyperplane and all training points

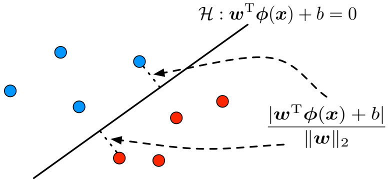
$$\text{MARGIN}(\boldsymbol{w}, b) = \min_n \frac{y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]}{\|\boldsymbol{w}\|_2}$$



Optimizing the Margin

Margin Smallest distance between the hyperplane and all training points

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

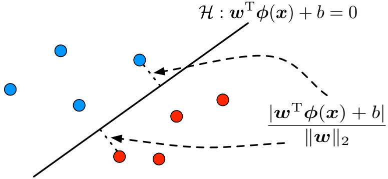


How should we pick (w, b) based on its margin?

Optimizing the Margin

Margin Smallest distance between the hyperplane and all training points

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



How should we pick (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!

$$\max_{\boldsymbol{w},b} \quad \min_n \frac{y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]}{\|\boldsymbol{w}\|} = \max_{\boldsymbol{w},b} \frac{1}{\|\boldsymbol{w}\|_2} \min_n y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]$$

Rescaled Margin

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

$$\min_n y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 1$$

Rescaled Margin

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

$$\min_n y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 1$$

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

$$\textsc{margin}(\boldsymbol{w}, b) = \frac{1}{\|\boldsymbol{w}\|_2}$$

Rescaled Margin

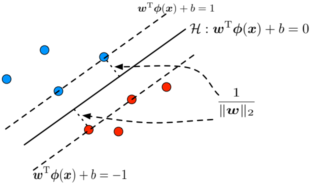
We can further constrain the problem by scaling $(\boldsymbol{w}, b)$ such that

$$\min_n y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 1$$

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

$$\textsc{margin}(\boldsymbol{w}, b) = \frac{1}{\|\boldsymbol{w}\|_2}$$

Hence the points closest to the decision boundary are at distance 1!



SVM: max margin formulation for separable data

Assuming separable training data, we thus want to solve:

$$\max_{\boldsymbol{w},b} \frac{1}{\|\boldsymbol{w}\|_2} \quad \text{such that} \quad y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1, \quad \forall \quad n$$

This is equivalent to

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2$$
$$\text{s.t.} \quad y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1, \quad \forall \quad n$$

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

SVM for non-separable data

Constraints in separable setting

$$y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1, \quad \forall \quad n$$

Constraints in non-separable setting

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

$$y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1 - \xi_n, \quad \forall \quad n$$

SVM for non-separable data

Constraints in separable setting

$$y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1, \quad \forall \quad n$$

Constraints in non-separable setting

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

$$y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1 - \xi_n, \quad \forall \quad n$$

- 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:

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

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

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

- C is user-defined regularization hyperparameter that trades off between the two terms in our objective
- This is a *convex quadratic program* that can be solved with general purpose or specialized solvers

What are support vectors?

Support vectors are data points where the margin inequality constraint is active (i.e., an equality):

$$1 - \xi_n - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]\} = 0$$

3 types of SVs

- $\xi_n = 0.$

What are support vectors?

Support vectors are data points where the margin inequality constraint is active (i.e., an equality):

$$1 - \xi_n - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]\} = 0$$

3 types of SVs

- $\xi_n = 0$. This implies $y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 1$. These are points that are on the margin
- $\xi_n < 1$.

What are support vectors?

Support vectors are data points where the margin inequality constraint is active (i.e., an equality):

$$1 - \xi_n - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]\} = 0$$

3 types of SVs

- $\xi_n = 0$. This implies $y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 1$. These are points that are on the margin
- $\xi_n < 1$. These are points that can be classified correctly but do not satisfy the large margin constraint, i.e., distance to the hyperplane is less than 1
- $\xi_n > 1$.

What are support vectors?

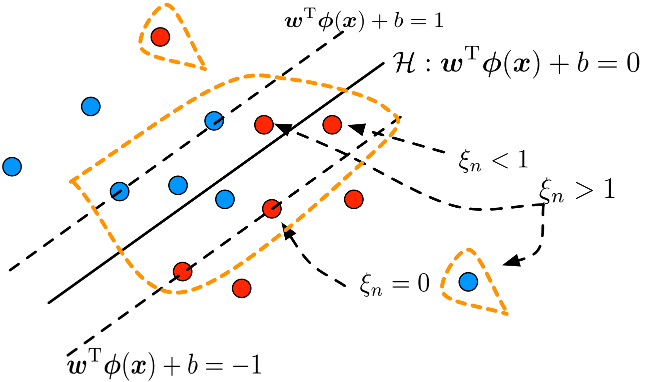
Support vectors are data points where the margin inequality constraint is active (i.e., an equality):

$$1 - \xi_n - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]\} = 0$$

3 types of SVs

- $\xi_n = 0$. This implies $y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 1$. These are points that are on the margin
- $\xi_n < 1$. These are points that can be classified correctly but do not satisfy the large margin constraint, i.e., distance to the hyperplane is less than 1
- $\xi_n > 1$. These are points that are misclassified.

Visualization of how training data points are categorized



- The SVM solution solution is only determined by a subset of the training samples (as we will see later in the lecture)
- These samples are called *support vectors*, which are highlighted by the dotted orange lines in the figure

Outline

Hinge loss

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

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

Intuition

Hinge loss

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

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

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

Hinge loss

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

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

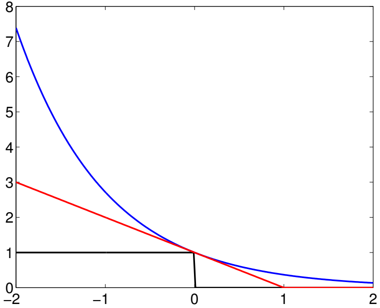
Intuition

- 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

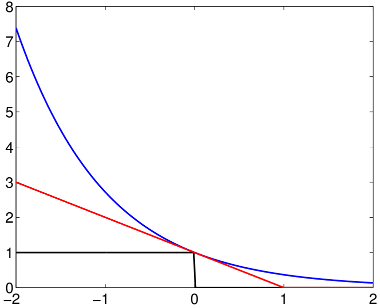
$$\ell^{\text{HINGE}}(f(\boldsymbol{x}), y) = \max(0, 1 - yf(\boldsymbol{x})) = (1 - yf(\boldsymbol{x}))_+$$

Visualization and Properties



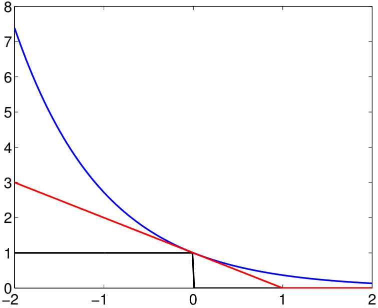
- Upper-bound for $0/1$ loss function (black line)
- We use hinge loss is a *surrogate* to $0/1$ loss – Why?

Visualization and Properties



- Upper-bound for $0/1$ loss function (black line)
- We use hinge loss is a *surrogate* to $0/1$ loss – Why?
- Hinge loss is convex, and thus easier to work with (though it's not differentiable at kink)

Visualization and Properties



- Other surrogate losses can be used, e.g., exponential loss for Adaboost (in blue), logistic loss (not shown) for logistic regression
- Hinge loss less sensitive to outliers than exponential (or logistic) loss
- Logistic loss has a natural probabilistic interpretation
- We can greedily optimize exponential loss (Adaboost)

Primal formulation of support vector machines (SVM)

Minimizing the total hinge loss on all the training data

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

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

Primal formulation of support vector machines (SVM)

Minimizing the total hinge loss on all the training data

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

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

Previously, we used geometric arguments to derive:

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

$$\text{s.t.} \quad y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1 - \xi_n \quad \text{and} \quad \xi_n \geq 0, \quad \forall\, n$$

Do these the yield the same solution?

Recovering our previous SVM formulation

Define $C = 1/\lambda$:

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

Recovering our previous SVM formulation

Define $C = 1/\lambda$:

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

Define $\xi_n \geq \max(0, 1 - y_n f(\boldsymbol{x}_n))$

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

$$\text{s.t.} \quad \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) + b]) \leq \xi_n, \quad \forall \; n$$

Since $c \geq \max(a, b) \iff c \geq a, \, c \geq b$, we recover previous formulation

Outline

Kernel SVM Roadmap

Key concepts we'll cover

- Brief review of constrained optimization with inequality constraints
  - "Primal" and "Dual" problems
  - Strong Duality and KKT conditions
- Dual SVM problem and Kernel SVM
- Dual SVM problem and support vectors

Constrained Optimization – Equality Constraints

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x})$$
$$\text{s.t.} \quad h_j(\boldsymbol{x}) = 0, \quad \forall\ j$$

The Lagrangian is defined as follows:

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

When problem is convex, we can find the optimal solution by

- Computing partial derivatives of L
- Setting them to zero
- Solving the corresponding system of equations

Constrained Optimization – Inequality Constraints

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

This is the 'primal' problem

Constrained Optimization – Inequality Constraints

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

This is the 'primal' problem with the *generalized* Lagrangian:

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

Constrained Optimization – Inequality Constraints

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

This is the 'primal' problem with the *generalized* Lagrangian:

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

Consider the following function:

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

Constrained Optimization – Inequality Constraints

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

This is the 'primal' problem with the *generalized* Lagrangian:

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

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 $\theta_P(\boldsymbol{x}) = f(\boldsymbol{x})$

Constrained Optimization – Inequality Constraints

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

This is the 'primal' problem with the *generalized* Lagrangian:

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

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

Constrained Optimization – Inequality Constraints

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

This is the 'primal' problem with the *generalized* Lagrangian:

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

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 primal problem, which we denote as p^*

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

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

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

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

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

Primal and dual are the same, except the max and min are exchanged!

Relationship between primal and dual?

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

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

Primal and dual are the same, except the max and min are exchanged!

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!

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

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!

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!

When these conditions hold, there must 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)

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)
- 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 holds for the SVM problem, and in particular the KKT conditions are necessary and sufficient for the optimal solution
- See `http://cs229.stanford.edu/notes/cs229-notes3.pdf` for details

Dual formulation of SVM

Dual is also a convex quadratic programming

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

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

$$\sum_n \alpha_n y_n = 0$$

Dual formulation of SVM

Dual is also a convex quadratic programming

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

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

$$\sum_n \alpha_n y_n = 0$$

- There are N dual variable α_n, one for each constraint in the primal formulation

Kernel SVM

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

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

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

$$\sum_n \alpha_n y_n = 0$$

We can define a kernel function to work with nonlinear features and learn a nonlinear decision surface

Recovering solution to the primal formulation

Weights

$$\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) \leftarrow \text{ Linear combination of the input features}$$

Recovering solution to the primal formulation

Weights

$$\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) \leftarrow \text{ Linear combination of the input features}$$

Offset For any $C > \alpha_n > 0$, we have

$$b = [y_n - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)]$$

Recovering solution to the primal formulation

Weights

$$\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) \leftarrow \text{ Linear combination of the input features}$$

Offset For any $C > \alpha_n > 0$, we have

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

Recovering solution to the primal formulation

Weights

$$\boldsymbol{w} = \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) \leftarrow \text{ Linear combination of the input features}$$

Offset For any $C > \alpha_n > 0$, we have

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

Prediction on a test point x

$$h(\boldsymbol{x}) = \mathrm{SIGN}(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}) + b) = \mathrm{SIGN}(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b)$$

At test time it suffices to know the kernel function!

Derivation of the dual

We will next derive the dual formulation for SVMs.

Recipe

- Formulate the generalized Lagrangian function that incorporates the constraints and introduces dual variables
- Minimize the Lagrangian function over the primal variables
- Substitute the primal variables for dual variables in the Lagrangian
- Maximize the Lagrangian with respect to dual variables
- Recover the solution (for the primal variables) from the dual variables

A simple example

Consider the example of convex quadratic programming

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

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)

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

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 $\alpha_2 \geq 0$.

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

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

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

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:

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

Example (cont'd)

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

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

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

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

Deriving the dual for SVM

Primal SVM

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

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

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

Deriving the dual for SVM

Primal SVM

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

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

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

Lagrangian

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

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

Minimizing the Lagrangian

Taking derivatives with respect to the primal variables

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) = 0$$

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

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

Minimizing the Lagrangian

Taking derivatives with respect to the primal variables

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_n y_n \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) = 0$$

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

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

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

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

$$\sum_n \alpha_n y_n = 0$$

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

Rearrange the Lagrangian and incorporate these constraints

Recall:

- $L(\cdot) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] - \xi_n\}$ where $\alpha_n \geq 0$ and $\lambda_n \geq 0$
- Constraints from partial derivatives: $\sum_n \alpha_n y_n = 0$ and $C - \lambda_n - \alpha_n = 0$

$$
\begin{aligned}
g(\{\alpha_n\},\{\lambda_n\}) &= L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) \\
&= \sum_n (C - \alpha_n - \lambda_n)\xi_n
\end{aligned}
$$

Rearrange the Lagrangian and incorporate these constraints

Recall:

- $L(\cdot) = C \sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n \xi_n + \sum_n \alpha_n \{1 - y_n[\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) + b] - \xi_n\}$ where $\alpha_n \geq 0$ and $\lambda_n \geq 0$
- Constraints from partial derivatives: $\sum_n \alpha_n y_n = 0$ and $C - \lambda_n - \alpha_n = 0$

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

Rearrange the Lagrangian and incorporate these constraints

Recall:

- $L(\cdot) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n\xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] - \xi_n\}$ where $\alpha_n \geq 0$ and $\lambda_n \geq 0$
- Constraints from partial derivatives: $\sum_n \alpha_n y_n = 0$ and $C - \lambda_n - \alpha_n = 0$

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

Rearrange the Lagrangian and incorporate these constraints

Recall:

- $L(\cdot) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n\xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] - \xi_n\}$ where $\alpha_n \geq 0$ and $\lambda_n \geq 0$
- Constraints from partial derivatives: $\sum_n \alpha_n y_n = 0$ and $C - \lambda_n - \alpha_n = 0$

$$
\begin{aligned}
g(\{\alpha_n\},\{\lambda_n\}) &= L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) \\
&= \sum_n (C - \alpha_n - \lambda_n)\xi_n + \frac{1}{2}\|\sum_n y_n\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 + \sum_n \alpha_n \\
&\quad - \left(\sum_n \alpha_n y_n\right) b - \sum_n \alpha_n y_n \left(\sum_m y_m\alpha_m\boldsymbol{\phi}(\boldsymbol{x}_m)\right)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n)
\end{aligned}
$$

Rearrange the Lagrangian and incorporate these constraints

Recall:

- $L(\cdot) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n\xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] - \xi_n\}$ where $\alpha_n \geq 0$ and $\lambda_n \geq 0$
- Constraints from partial derivatives: $\sum_n \alpha_n y_n = 0$ and $C - \lambda_n - \alpha_n = 0$

$$
\begin{aligned}
g(\{\alpha_n\},\{\lambda_n\}) &= L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) \\
&= \sum_n (C - \alpha_n - \lambda_n)\xi_n + \frac{1}{2}\|\sum_n y_n\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 + \sum_n \alpha_n \\
&\quad - \left(\sum_n \alpha_n y_n\right) b - \sum_n \alpha_n y_n \left(\sum_m y_m\alpha_m\boldsymbol{\phi}(\boldsymbol{x}_m)\right)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \\
&= \sum_n \alpha_n + \frac{1}{2}\|\sum_n y_n\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 - \sum_{m,n} \alpha_n\alpha_m y_m y_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)
\end{aligned}
$$

Rearrange the Lagrangian and incorporate these constraints

Recall:

- $L(\cdot) = C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n\xi_n + \sum_n \alpha_n\{1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] - \xi_n\}$ where $\alpha_n \geq 0$ and $\lambda_n \geq 0$
- Constraints from partial derivatives: $\sum_n \alpha_n y_n = 0$ and $C - \lambda_n - \alpha_n = 0$

$$
\begin{aligned}
g(\{\alpha_n\},\{\lambda_n\}) &= L(\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}) \\
&= \sum_n (C - \alpha_n - \lambda_n)\xi_n + \frac{1}{2}\|\sum_n y_n\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 + \sum_n \alpha_n \\
&- \left(\sum_n \alpha_n y_n\right) b - \sum_n \alpha_n y_n \left(\sum_m y_m\alpha_m\boldsymbol{\phi}(\boldsymbol{x}_m)\right)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \\
&= \sum_n \alpha_n + \frac{1}{2}\|\sum_n y_n\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 - \sum_{m,n} \alpha_n\alpha_m y_m y_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) \\
&= \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} \alpha_n\alpha_m y_m y_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)
\end{aligned}
$$

The dual problem

Maximizing the dual under the constraints

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

The dual problem

Maximizing the dual under the constraints

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

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

$$\sum_n \alpha_n y_n = 0$$

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

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

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: $\alpha_n \leq C$:

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

Simplified Dual

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

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

$$\sum_n \alpha_n y_n = 0$$

Recovering solution to the primal formulation

We already identified the primal variable: $\frac{\partial L}{\partial \boldsymbol{w}} \rightarrow \boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{\phi}(\boldsymbol{x}_n)$

Recovering solution to the primal formulation

We already identified the primal variable: $\frac{\partial L}{\partial \boldsymbol{w}} \rightarrow \boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{\phi}(\boldsymbol{x}_n)$

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

When does $\alpha_n > 0$?

- KKT conditions tell us:

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

Recovering solution to the primal formulation

We already identified the primal variable: $\frac{\partial L}{\partial \boldsymbol{w}} \rightarrow \boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{\phi}(\boldsymbol{x}_n)$

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

When does $\alpha_n > 0$?

- KKT conditions tell us:

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

- (2) tells us that $\alpha_n > 0$ iff $1 - \xi_n = y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]$
  - If $\xi_n = 0$, then support vector is on the margin
  - Otherwise, $\xi_n > 0$ means that the point is an outlier
- Equality from derivative of Lagrangian yields: $(3)\ C - \alpha_n - \lambda_n = 0$
  - If $\xi_n > 0$, then (1) and (3) imply that $\alpha_n = C$

Expressions for offset (b) and test predictions

Recovering b

- KKT conditions:

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

- Equality from derivative of Lagrangian: $(3)\ C - \alpha_n - \lambda_n = 0$

Combine (1) and (3) and assume $\alpha_n < C$:

Expressions for offset (b) and test predictions

Recovering b

- KKT conditions:

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

- Equality from derivative of Lagrangian: $(3)\ C - \alpha_n - \lambda_n = 0$

Combine (1) and (3) and assume $\alpha_n < C$:

$$\lambda_n = C - \alpha_n > 0 \rightarrow \xi_n = 0$$

Using (2), if $0 < \alpha_n < C$ and $y_n \in \{-1, 1\}$:

$$1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 0 \rightarrow b = y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)$$

Expressions for offset (b) and test predictions

Recovering b

- KKT conditions:

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

- Equality from derivative of Lagrangian: $(3)\ C - \alpha_n - \lambda_n = 0$

Combine (1) and (3) and assume $\alpha_n < C$:

$$\lambda_n = C - \alpha_n > 0 \rightarrow \xi_n = 0$$

Using (2), if $0 < \alpha_n < C$ and $y_n \in \{-1, 1\}$:

$$1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 0 \rightarrow b = y_n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)$$

Test Prediction: $h(\boldsymbol{x}) = \mathrm{SIGN}(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b)$