CSCI567 Machine Learning (Fall 2020)

Prof. Haipeng Luo

U of Southern California

Oct 1, 2020

Administration

HW 2 grade will be released by 10/06. Solutions will be discussed today.

Quiz 1 on 10/08:

- Coverage: **mostly Lec 1-5**, 1-2 multiple choice questions from Lec 6.
- Join the usual zoom meeting 5-10 mins earlier; will be assigned to a breakout room, proctored by a TA/CP with your **camera on**.
- At 4:55pm, Crowdmark will send you the quiz automatically.
- Open-book/note, but *no collaboration or consultation*.
- For multiple choice, select **one and only one answer**.
- Upload answers for each question, just like HW.
- Duration is 2.5 hours, which *includes the time for scanning/uploading*.

Outline

1. Review of last lecture
2. Support vector machines (primal formulation)
3. A detour of Lagrangian duality
4. Support vector machines (dual formulation)

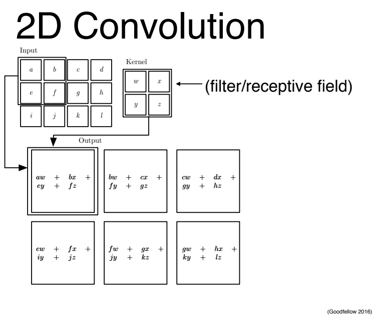
Outline

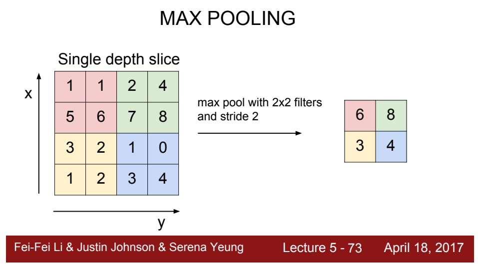
1. Review of last lecture
2. Support vector machines (primal formulation)
3. A detour of Lagrangian duality
4. Support vector machines (dual formulation)

Convolutional Neural Nets

Typical architecture for CNNs:

Input $\to$ [[Conv $\to$ ReLU]*N $\to$ Pool?]*M $\to$ [FC $\to$ ReLU]*Q $\to$ FC





Kernel functions

Definition: a function $k : \mathbb{R}^{\mathsf{D}} \times \mathbb{R}^{\mathsf{D}} \to \mathbb{R}$ is called a *(positive semidefinite) kernel function* if there exists a function $\boldsymbol{\phi} : \mathbb{R}^{\mathsf{D}} \to \mathbb{R}^{\mathsf{M}}$ so that for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{\mathsf{D}}$,

$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}')$$

Examples we have seen

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^{\mathrm{T}} \boldsymbol{x}')^2$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{\mathsf{D}} \frac{\sin(2\pi(x_d - x'_d))}{x_d - x'_d}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^{\mathrm{T}} \boldsymbol{x}' + c)^d \qquad \textbf{(polynomial kernel)}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = e^{-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2}{2\sigma^2}} \qquad \textbf{(Gaussian/RBF kernel)}$$

Kernelizing ML algorithms

Feasible as long as **only inner products are required**:

- regularized linear regression (dual formulation)

$$\boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{w}^* = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{\Phi}^{\mathrm{T}} (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y} \quad (\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}} \text{ is } \textit{kernel matrix})$$

- nearest neighbor classifier with L2 distance

$$\|\boldsymbol{\phi}(\boldsymbol{x}) - \boldsymbol{\phi}(\boldsymbol{x}')\|_2^2 = k(\boldsymbol{x}, \boldsymbol{x}) + k(\boldsymbol{x}', \boldsymbol{x}') - 2k(\boldsymbol{x}, \boldsymbol{x}')$$

- perceptron, logistic regression, SVM, . . .

Outline

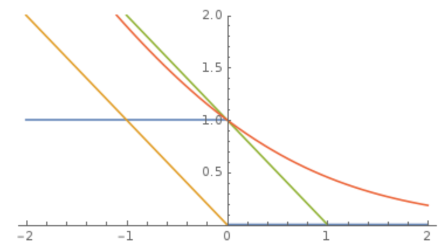
Support vector machines (SVM)

- One of the most commonly used classification algorithms
- Works well with the kernel trick
- Strong theoretical guarantees

We focus on **binary classification** here.

Primal formulation

In one sentence: linear model with L2 regularized hinge loss. Recall



- perceptron loss $\ell_{\text{perceptron}}(z) = \max\{0, -z\} \rightarrow$ Perceptron
- logistic loss $\ell_{\text{logistic}}(z) = \log(1 + \exp(-z)) \rightarrow$ logistic regression
- hinge loss $\ell_{\text{hinge}}(z) = \max\{0, 1 - z\} \rightarrow$ **SVM**

Primal formulation

For a linear model $(\boldsymbol{w}, b)$, this means

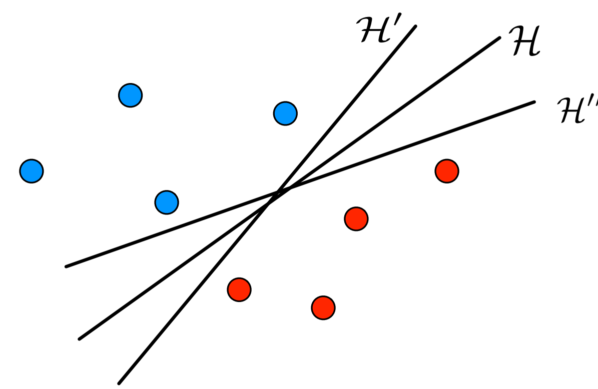
$$\min_{\boldsymbol{w},b} \sum_n \max\left\{0, 1 - y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b)\right\} + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

- recall $y_n \in \{-1, +1\}$
- a nonlinear mapping $\boldsymbol{\phi}$ is applied
- the bias/intercept term b is used explicitly (think about why after this lecture)

So why L2 regularized hinge loss?

Geometric motivation: separable case

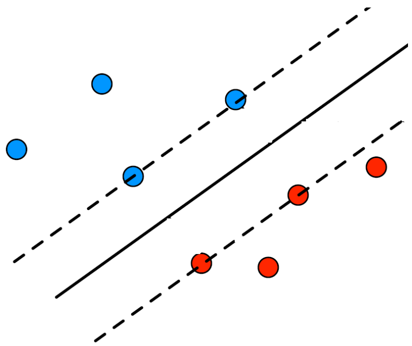
When data is **linearly separable**, there are *infinitely many hyperplanes with zero training error*:



So which one should we choose?

Intuition

The further away from data points the better.



How to formalize this intuition?

Distance to hyperplane

What is the **distance** from a point $\boldsymbol{x}$ to a hyperplane $\{\boldsymbol{x} : \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0\}$?

Assume the **projection** is $\boldsymbol{x} - \ell\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}$, then

$$0 = \boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{x} - \ell\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}\right) + b = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} - \ell\|\boldsymbol{w}\| + b$$

and thus $\ell = \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}{\|\boldsymbol{w}\|_2}$.

Therefore the distance is

$$\frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b|}{\|\boldsymbol{w}\|_2}$$

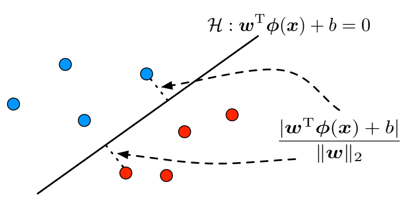
For a hyperplane that correctly classifies $(\boldsymbol{x}, y)$, the distance becomes

$$\frac{y(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b)}{\|\boldsymbol{w}\|_2}$$

Maximizing margin

Margin: the *smallest* distance from all training points to the hyperplane

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



The intuition "**the further away the better**" translates to solving

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

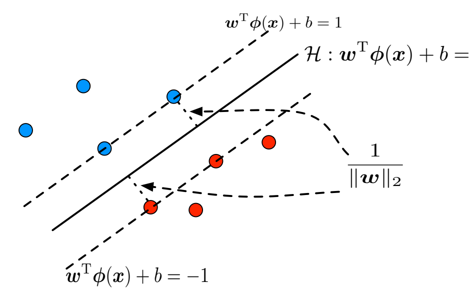
Rescaling

Note: rescaling $(\boldsymbol{w}, b)$ does not change the hyperplane at all.

We can thus always scale $(\boldsymbol{w}, b)$ s.t. $\min_n y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) = 1$

The margin then becomes

$$\begin{aligned}&\text{MARGIN OF } (\boldsymbol{w},\, b)\\ &= \frac{1}{\|\boldsymbol{w}\|_2} \min_n y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b)\\ &= \frac{1}{\|\boldsymbol{w}\|_2}\end{aligned}$$



Summary for separable data

For a separable training set, we aim to solve

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

This is equivalent to

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

SVM is thus also called *max-margin* classifier. The constraints above are called *hard-margin* constraints.

General non-separable case

If data is not linearly separable, the previous constraint

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

is obviously *not feasible*.

To deal with this issue, we relax them to **soft-margin** constraints:

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

where we introduce **slack variables** $\xi_n \geq 0$.

SVM Primal formulation

We want ξ_n to be as small as possible too. The objective becomes

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

where C is a hyperparameter to balance the two goals.

Equivalent form

Formulation

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

is equivalent to

$$\begin{aligned} \min_{\boldsymbol{w},b,\{\xi_n\}} \quad & C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \\ \text{s.t.} \quad & \max\left\{0, 1 - y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)+b)\right\} = \xi_n, \quad \forall\ n \end{aligned}$$

Equivalent form

$$\min_{\boldsymbol{w},b,\{\xi_n\}} \quad C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2$$
$$\text{s.t.} \quad \max\left\{0, 1 - y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b)\right\} = \xi_n, \quad \forall\ n$$

is equivalent to

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

and

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

with $\lambda = 1/C$. *This is exactly minimizing L2 regularized hinge loss!*

Optimization

$$\min_{\boldsymbol{w},b,\{\xi_n\}} \quad C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2$$
$$\text{s.t.} \quad 1 - y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) \le \xi_n, \quad \forall\ n$$
$$\xi_n \ge 0, \quad \forall\ n$$

- It is a convex (**quadratic** in fact) problem
- thus can apply any convex optimization algorithms, e.g. SGD
- there are **more specialized and efficient** algorithms
- but usually we apply kernel trick, which requires solving the *dual problem*

Outline

Lagrangian duality

Extremely important and powerful tool in analyzing optimizations

We will introduce basic concepts and derive the **KKT conditions**

Applying it to SVM reveals an important aspect of the algorithm

Primal problem

Suppose we want to solve

$$\min_{\boldsymbol{w}} F(\boldsymbol{w}) \quad \text{s.t. } h_j(\boldsymbol{w}) \leq 0 \quad \forall\, j \in [\mathsf{J}]$$

where functions $h_1, \ldots, h_{\mathsf{J}}$ define J **constraints**.

SVM primal formulation is clearly of this form with $\mathsf{J} = 2\mathsf{N}$ constraints:

$$F(\boldsymbol{w}, b, \{\xi_n\}) = C \sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2$$
$$h_n(\boldsymbol{w}, b, \{\xi_n\}) = 1 - y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) - \xi_n \quad \forall\, n \in [\mathsf{N}]$$
$$h_{\mathsf{N}+n}(\boldsymbol{w}, b, \{\xi_n\}) = -\xi_n \quad \forall\, n \in [\mathsf{N}]$$

Lagrangian

The **Lagrangian** of the previous problem is defined as:

$$L(\boldsymbol{w}, \{\lambda_j\}) = F(\boldsymbol{w}) + \sum_{j=1}^{\mathsf{J}} \lambda_j h_j(\boldsymbol{w})$$

where $\lambda_1, \ldots, \lambda_{\mathsf{J}} \geq 0$ are called **Lagrangian multipliers**.

Note that

$$\max_{\{\lambda_j\}\geq 0} L(\boldsymbol{w}, \{\lambda_j\}) = \begin{cases} F(\boldsymbol{w}) & \text{if } h_j(\boldsymbol{w}) \leq 0 \quad \forall\, j \in [\mathsf{J}] \\ +\infty & \text{else} \end{cases}$$

and thus,

$$\min_{\boldsymbol{w}} \max_{\{\lambda_j\}\geq 0} L(\boldsymbol{w}, \{\lambda_j\}) \iff \min_{\boldsymbol{w}} F(\boldsymbol{w}) \ \text{ s.t. } h_j(\boldsymbol{w}) \leq 0 \quad \forall\, j \in [\mathsf{J}]$$

Duality

We define the **dual problem** by swapping the min and max:

$$\max_{\{\lambda_j\}\geq 0} \min_{\boldsymbol{w}} L(\boldsymbol{w}, \{\lambda_j\})$$

How are the primal and dual connected? Let $\boldsymbol{w}^*$ and $\{\lambda_j^*\}$ be the primal and dual solutions respectively, then

$$\max_{\{\lambda_j\}\geq 0} \min_{\boldsymbol{w}} L(\boldsymbol{w}, \{\lambda_j\}) = \min_{\boldsymbol{w}} L\left(\boldsymbol{w}, \{\lambda_j^*\}\right) \leq L\left(\boldsymbol{w}^*, \{\lambda_j^*\}\right)$$
$$\leq \max_{\{\lambda_j\}\geq 0} L(\boldsymbol{w}^*, \{\lambda_j\}) = \min_{\boldsymbol{w}} \max_{\{\lambda_j\}\geq 0} L(\boldsymbol{w}, \{\lambda_j\})$$

This is called "**weak duality**".

Strong duality

When $F, h_1, \ldots, h_m$ are convex, under some mild conditions:

$$\min_{\boldsymbol{w}} \max_{\{\lambda_j\}\geq 0} L(\boldsymbol{w}, \{\lambda_j\}) = \max_{\{\lambda_j\}\geq 0} \min_{\boldsymbol{w}} L(\boldsymbol{w}, \{\lambda_j\})$$

This is called "**strong duality**".

Deriving the Karush-Kuhn-Tucker (KKT) conditions

Observe that if strong duality holds:

$$F(\boldsymbol{w}^*) = \min_{\boldsymbol{w}} \max_{\{\lambda_j\}\geq 0} L(\boldsymbol{w}, \{\lambda_j\}) = \max_{\{\lambda_j\}\geq 0} \min_{\boldsymbol{w}} L(\boldsymbol{w}, \{\lambda_j\})$$

$$= \min_{\boldsymbol{w}} L(\boldsymbol{w}, \{\lambda_j^*\}) \leq L(\boldsymbol{w}^*, \{\lambda_j^*\}) = F(\boldsymbol{w}^*) + \sum_{j=1}^{\mathsf{J}} \lambda_j^* h_j(\boldsymbol{w}^*) \leq F(\boldsymbol{w}^*)$$

Implications:

- *all inequalities above have to be equalities!*
- last equality implies $\lambda_j^* h_j(\boldsymbol{w}^*) = 0$ for all $j \in [\mathsf{J}]$
- equality $\min_{\boldsymbol{w}} L(\boldsymbol{w}, \{\lambda_j^*\}) = L(\boldsymbol{w}^*, \{\lambda_j^*\})$ implies $\boldsymbol{w}^*$ is a **minimizer** of $L(\boldsymbol{w}, \{\lambda_j^*\})$ and thus has **zero gradient**:

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}^*, \{\lambda_j^*\}) = \nabla F(\boldsymbol{w}^*) + \sum_{j=1}^{\mathsf{J}} \lambda_j^* \nabla h_j(\boldsymbol{w}^*) = \mathbf{0}$$

The Karush-Kuhn-Tucker (KKT) conditions

If $\boldsymbol{w}^*$ and $\{\lambda_j^*\}$ are the primal and dual solution respectively, then:

Stationarity:

$$\nabla_{\boldsymbol{w}} L(\boldsymbol{w}^*, \{\lambda_j^*\}) = \nabla F(\boldsymbol{w}^*) + \sum_{j=1}^{\mathsf{J}} \lambda_j^* \nabla h_j(\boldsymbol{w}^*) = \mathbf{0}$$

Complementary slackness:

$$\lambda_j^* h_j(\boldsymbol{w}^*) = 0 \quad \text{for all } j \in [\mathsf{J}]$$

Feasibility:

$$h_j(\boldsymbol{w}^*) \leq 0 \quad \text{and} \quad \lambda_j^* \geq 0 \quad \text{for all } j \in [\mathsf{J}]$$

These are *necessary conditions*. They are also *sufficient* when F is convex and $h_1, \ldots, h_{\mathsf{J}}$ are continuously differentiable convex functions.

Outline

Writing down the Lagrangian

Recall the primal formulation

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

Lagrangian is

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

where $\alpha_1, \ldots, \alpha_{\mathsf{N}} \geq 0$ and $\lambda_1, \ldots, \lambda_{\mathsf{N}} \geq 0$ are Lagrangian multipliers.

Applying the stationarity condition

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

$\exists$ primal and dual variables $\boldsymbol{w}, b, \{\xi_n\}, \{\alpha_n\}, \{\lambda_n\}$ s.t. $\nabla_{\boldsymbol{w},b,\{\xi_n\}} L = \boldsymbol{0}$, which means

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

$$\frac{\partial L}{\partial b} = -\sum_n \alpha_n y_n = 0 \quad \text{and} \quad \frac{\partial L}{\partial \xi_n} = C - \lambda_n - \alpha_n = 0, \quad \forall\, n$$

Rewrite the Lagrangian in terms of dual variables

Replacing $\boldsymbol{w}$ by $\sum_n y_n\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)$ in the Lagrangian gives

$$\begin{aligned}
L &= C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \sum_n \lambda_n\xi_n + \sum_n \alpha_n\left(1 - y_n(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b) - \xi_n\right) \\
&= C\sum_n \xi_n + \frac{1}{2}\|\sum_n y_n\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 - \sum_n \lambda_n\xi_n + \\
&\quad \sum_n \alpha_n\left(1 - y_n\left(\left(\sum_m y_m\alpha_m\boldsymbol{\phi}(\boldsymbol{x}_m)\right)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b\right) - \xi_n\right) \\
&= \sum_n \alpha_n + \frac{1}{2}\|\sum_n y_n\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n)\|_2^2 - \sum_{m,n} \alpha_n\alpha_m y_m y_n\boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) \\
&\qquad\qquad (\textstyle\sum_n \alpha_n y_n = 0 \text{ and } C = \lambda_n + \alpha_n) \\
&= \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} \alpha_n\alpha_m y_m y_n\boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n)
\end{aligned}$$

The dual formulation

To find the dual solutions, it amounts to solving

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

Note the last three constraints can be written as $0 \leq \alpha_n \leq C$ for all n. So the final **dual formulation of SVM** is:

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

Kernelizing SVM

Now it is clear that with a **kernel function** k for the mapping ϕ, we can kernelize SVM as:

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

Again, no need to compute $\boldsymbol{\phi}(\boldsymbol{x})$. It is a **quadratic program** and many efficient optimization algorithms exist.

Recover the primal solution

But how do we predict given the dual solution $\{\alpha_n^*\}$? Need to figure out the primal solution $\boldsymbol{w}^*$ and b^*.

Based on previous observation,

$$\boldsymbol{w}^* = \sum_n \alpha_n^* y_n \boldsymbol{\phi}(\boldsymbol{x}_n) = \sum_{n:\alpha_n>0} \alpha_n^* y_n \boldsymbol{\phi}(\boldsymbol{x}_n)$$

A point with $\alpha_n^* > 0$ is called a "**support vector**". Hence the name SVM.

To identify b, we need to apply complementary slackness.

Applying complementary slackness

For all n we should have

$$\lambda_n^* \xi_n^* = 0, \quad \alpha_n^* \left(1 - \xi_n^* - y_n(\boldsymbol{w}^{*\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) + b^*)\right) = 0$$

For any support vector $\boldsymbol{\phi}(\boldsymbol{x}_n)$ with $0 < \alpha_n^* < C$, $\lambda_n^* = C - \alpha_n^* > 0$ holds.

- first condition implies $\xi_n^* = 0$.
- second condition implies $1 = y_n(\boldsymbol{w}^{*\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) + b^*)$ and thus

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

Usually *average* over all n with $0 < \alpha_n^* < C$ to stabilize computation.

The prediction on a new point $\boldsymbol{x}$ is therefore

$$\mathrm{SGN}\left(\boldsymbol{w}^{*\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}) + b^*\right) = \mathrm{SGN}\left(\sum_m y_m \alpha_m^* k(\boldsymbol{x}_m, \boldsymbol{x}) + b^*\right)$$

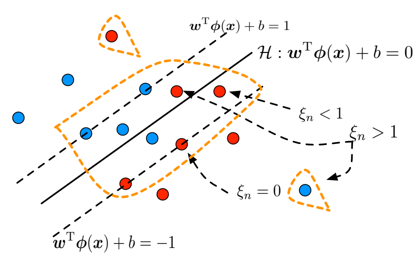
Geometric interpretation of support vectors

A support vector satisfies $\alpha_n^* \neq 0$ and

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

When

- $\xi_n^* = 0$, $y_n(\boldsymbol{w}^{*\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) + b^*) = 1$ and thus the point is $1/\|\boldsymbol{w}^*\|_2$ away from the hyperplane.
- $\xi_n^* < 1$, the point is classified correctly but does not satisfy the large margin constraint.
- $\xi_n^* > 1$, the point is misclassified.

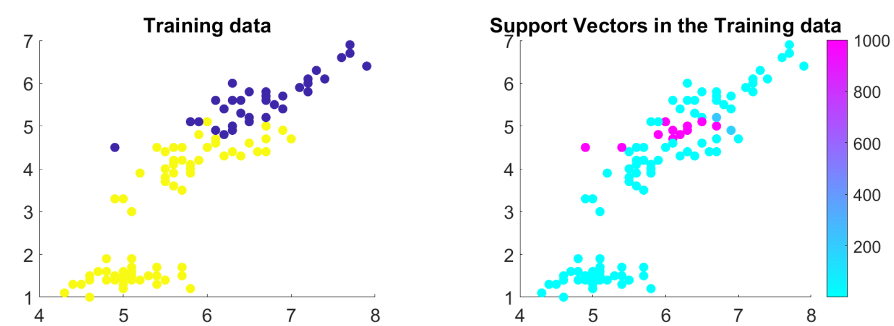


Support vectors (circled with the orange line) are *the only points that matter!*

An example

One drawback of kernel method: **non-parametric**, need to keep all training points potentially

For SVM, very often #support vectors $\ll$ N



Summary

SVM: **max-margin linear classifier**

Primal (equivalent to minimizing L2 regularized hinge loss):

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

Dual (kernelizable, reveals what training points are support vectors):

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

Summary

Typical steps of applying Lagrangian duality

- start with a primal problem
- write down the Lagrangian (one dual variable per constraint)
- apply KKT conditions to find the connections between primal and dual solutions
- eliminate primal variables and arrive at the dual formulation
- maximize the Lagrangian with respect to dual variables
- recover the primal solutions from the dual solutions