Statistical Computing Final

106033233 資工21 周聖諺

2021/06/15

Outline

- Dataset & EDA
- SMO
- Fourier Kernel Approximation
- Evaluation

Sequential Minimal Optimization(SMO)

SMO

Step 1. Select & Update

Select 2 variables α_i, α_j and update

Step 2. Box Constraint

Clip the value of α_j with complementary slackness

Derive the new values α_i^*, α_j^*

Step 3. Update Bias

Derive new bias b^* from α_i^*, α_j^*

SMO - Step 1. Select & Update

Denote x_i, y_i as i-th data point and label. Let $K_{i,j} = k(x_i, x_j)$, where $k(a, b)$ is the kernel function and $f_\phi(x_i)$ is the prediction function.

$$E_i = f(x_i) - y_i,\ E_j = f(x_j) - y_j$$

$$\eta = K_{i,i} + K_{j,j} - 2K_{i,j}$$

Then, we get a new value of α_j

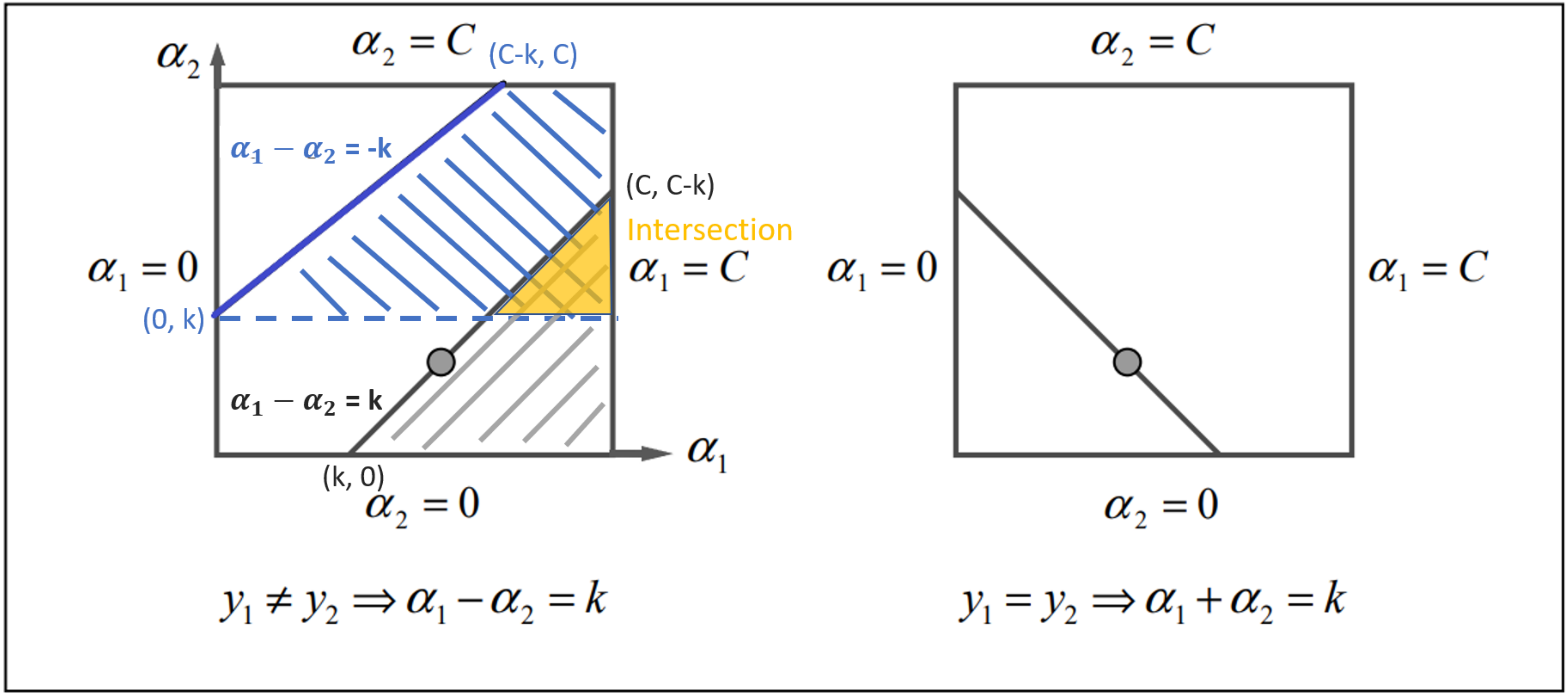
$$\alpha_j^{new} = \alpha_j + \frac{y_j(E_i - E_j)}{\eta}$$

To understand intuitively, you can see η as **Learning Rate** and $y_j(E_i - E_j)$ as a kind of **Loss**.

For more detailed derivation, please refer to the report.

SMO - Step 2. Box Constraint

To satisfy the complementary slackness $\alpha_1 y_1 + \alpha_2 y_2 = \zeta,\ 0 \le \alpha_i \le C$, we need to **clip** the α_j^{new} under blue and grey area.



SMO - Step 2. Box Constraint

- if($y_i = y_j$):
  - $B_U = \min(C, \alpha_j + \alpha_i),\ B_L = \max(0, \alpha_j + \alpha_i - C)$
- else:
  - $B_U = \min(C, C + \alpha_j - \alpha_i),\ B_L = \max(0, \alpha_j - \alpha_i)$
- $\alpha_j^* = CLIP(\alpha_j^{new}, B_L, B_U)$
- $\alpha_i^* = \alpha_i + y_i y_j (\alpha_j - \alpha_j^*)$

SMO - Step 3. Update Bias

When $0 < \alpha_i^* < C$, the data point x_i is right on the margin such that $f_\phi(x) = y_i$.

- $b_i^* = -E_i - y_i K_{i,i}(\alpha_i^* - \alpha_i) - y_j K_{j,i}(\alpha_j^* - \alpha_j) + b$
- $b_j^* = -E_j - y_i K_{i,j}(\alpha_i^* - \alpha_i) - y_j K_{j,j}(\alpha_j^* - \alpha_j) + b$
- if($0 \leq \alpha_i \leq C$):
  - $b^* = b_i^*$
- else if($0 \leq \alpha_j \leq C$):
  - $b^* = b_j^*$
- else:
  - $b^* = \frac{b_i^* + b_j^*}{2}$

Fourier Kernel Approximation

Fourier Kernel Approximation

Based on the paper **Random Features for Large-Scale Kernel Machines** on NIPS'07

For a **shift-invariant kernel** $k(\delta)$, **Bochner's theorem** guarantees that its **Fourier transform** $p(\omega)$ is a **probability distribution**. Defining $\zeta_\omega(x) = e^{j\omega' x}$, we have

$$k(x-y) = \int_\omega p(\omega) e^{j\omega'(x-y)} d\omega = E_\omega[\zeta_\omega(x)\zeta_\omega(y)]$$

where $\zeta_\omega(x)\zeta_\omega(y)$ is an unbiased estimate of $k(x,y)$ when ω is drawn from $p(\omega)$.

Time complexity: $\mathcal{O}(SN^3)$ with S samples. **Speed up the kernel computation with extremely large dimension.**

Fourier Kernel Approximation

Thus, to approximate **RBF kernel** with Monte-Carlo

$$K_{x,y} = z(x)'z(y) = \frac{1}{D}\sum_{j=1}^{D} z_{w_j}(x) z_{w_j}(y)$$

$$z_\omega(x) = \sqrt{2}cos(\omega x + b) \text{ where } \omega \sim p(\omega) = \mathcal{N}(0,1)$$

But when I apply the approximation to the dataset, it is still **not fast enough**. It may need GPU to speed up.