Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Optimizing Support Vector Machines

Jordan Boyd-Graber
University of Colorado Boulder
LECTURE 10

Slides adapted from David Page

Plan

Lagrange Multipliers

Introduce Lagrange variables $\alpha_i \geq 0$, $i \in [1, m]$ for each of the m constraints (one for each data point).

$$\mathscr{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i \left[y_i(w \cdot x_i + b) - 1 \right] \tag{1}$$

Lagrange Multipliers

Introduce Lagrange variables $\alpha_i \geq 0$, $i \in [1, m]$ for each of the m constraints (one for each data point).

$$\mathscr{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i \left[y_i(w \cdot x_i + b) - 1\right] \tag{1}$$

If $\alpha \neq 0$, then $y_i(w \cdot x_i + b) = \pm 1$.

Solving Lagrangian

Weights

$$\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \tag{2}$$

Solving Lagrangian

Weights

$$\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \tag{2}$$

Bias

$$0 = \sum_{i=1}^{m} \alpha_i y_i \tag{3}$$

Solving Lagrangian

Weights

$$\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \tag{2}$$

Bias

$$0 = \sum_{i=1}^{m} \alpha_i y_i \tag{3}$$

Support Vector-ness

$$\alpha_i = 0 \vee y_i(w \cdot x_i + b) = 1 \tag{4}$$

Reparameterize in terms of α

$$\max_{\vec{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{i=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j) \tag{5}$$

Outline for SVM Optimization (SMO)

1. Select two examples i, j
2. Get a learning rate η
3. Update α_j
4. Update α_i

Plan

Contrast with SG

- There's a learning rate η that depends on the data
- Use the error of an example to derive update
- You update multiple α at once

Contrast with SG

- There's a learning rate η that depends on the data
- Use the error of an example to derive update
- You update multiple α at once: if one goes up, the other should go down because $\sum y_i \alpha_i = 0$

More details

- We enforce every $\alpha_i < C$ (slackness)
- How do we know we've converged?

More details

- We enforce every $\alpha_i < C$ (slackness)
- How do we know we've converged?

$$\alpha_i = 0 \Rightarrow \quad y_i(w \cdot x_i + b) \geq 1 \tag{6}$$
$$\alpha_i = C \Rightarrow \quad y_i(w \cdot x_i + b) \leq 1 \tag{7}$$
$$0 < \alpha_i < C \Rightarrow \quad y_i(w \cdot x_i + b) = 1 \tag{8}$$

(Karush-Kuhn-Tucker Conditions)

More details

- We enforce every $\alpha_i < C$ (slackness)
- How do we know we've converged?

$$\alpha_i = 0 \Rightarrow \quad y_i(w \cdot x_i + b) \geq 1 \tag{6}$$

$$\alpha_i = C \Rightarrow \quad y_i(w \cdot x_i + b) \leq 1 \tag{7}$$

$$0 < \alpha_i < C \Rightarrow \quad y_i(w \cdot x_i + b) = 1 \tag{8}$$

(Karush-Kuhn-Tucker Conditions)

- Keep checking (to some tolerance)

Plan

Step 1: Select i and j

- Iterate over $i = \{1, \dots m\}$
- Repeat until KKT conditions are met
- Choose j randomly from $m - 1$ other options
- You can do better (particularly for large datasets)

Step 2: Optimize α_j

1. Compute upper (H) and lower (L) bounds that ensure $0 < \alpha_j \leq C$.

$y_i \neq y_j$

$$L = \max(0, \alpha_j - \alpha_i) \tag{9}$$

$$H = \min(C, C + \alpha_j - \alpha_i) \tag{10}$$

$y_i = y_j$

$$L = \max(0, \alpha_i + \alpha_j - C) \tag{11}$$

$$H = \min(C, \alpha_j + \alpha_i) \tag{12}$$

Step 2: Optimize α_j

1. Compute upper (H) and lower (L) bounds that ensure $0 < \alpha_j \leq C$.

$y_i \neq y_j$

$$L = \max(0, \alpha_j - \alpha_i) \tag{9}$$

$$H = \min(C, C + \alpha_j - \alpha_i) \tag{10}$$

$y_i = y_j$

$$L = \max(0, \alpha_i + \alpha_j - C) \tag{11}$$

$$H = \min(C, \alpha_j + \alpha_i) \tag{12}$$

This is because the update for α_i is based on $y_i y_j$ (sign matters)

Step 2: Optimize α_j

Compute errors for i and j

$$E_k \equiv f(x_k) - y_k \tag{13}$$

Step 2: Optimize α_j

Compute errors for i and j

$$E_k \equiv f(x_k) - y_k \tag{13}$$

and the learning rate (more similar, higher step size)

$$\eta = 2x_i \cdot x_j - x_i \cdot x_i - x_j \cdot x_j \tag{14}$$

Step 2: Optimize α_j

Compute errors for i and j

$$E_k \equiv f(x_k) - y_k \tag{13}$$

and the learning rate (more similar, higher step size)

$$\eta = 2x_i \cdot x_j - x_i \cdot x_i - x_j \cdot x_j \tag{14}$$

for new value for α_j

$$\alpha_j^* = \alpha_j - \frac{y_j(E_i - E_j)}{\eta} \tag{15}$$

Step 2: Optimize α_j

Compute errors for i and j

$$E_k \equiv f(x_k) - y_k \tag{13}$$

and the learning rate (more similar, higher step size)

$$\eta = 2x_i \cdot x_j - x_i \cdot x_i - x_j \cdot x_j \tag{14}$$

for new value for α_j

$$\alpha_j^* = \alpha_j - \frac{y_j(E_i - E_j)}{\eta} \tag{15}$$

Similar to stochastic gradient, but with additional error term.

Step 2: Optimize α_j

Compute errors for i and j

$$E_k \equiv f(x_k) - y_k \tag{13}$$

and the learning rate (more similar, higher step size)

$$\eta = 2x_i \cdot x_j - x_i \cdot x_i - x_j \cdot x_j \tag{14}$$

for new value for α_j

$$\alpha_j^* = \alpha_j - \frac{y_j(E_i - E_j)}{\eta} \tag{15}$$

Similar to stochastic gradient, but with additional error term. If α_j^* is outside $[L, H]$, clip it so that it is within the range.

Step 3: Optimize α_i

Set α_i:

$$\alpha_i^* = \alpha_i + y_i y_j \left(\alpha_j^{(old)} - \alpha_j \right) \tag{16}$$

Step 3: Optimize α_i

Set α_i:

$$\alpha_i^* = \alpha_i + y_i y_j \left(\alpha_j^{(old)} - \alpha_j \right) \tag{16}$$

This balances out the move that we made for α_j.

Step 4: Optimize the threshold b

We need the KKT conditions to be satisfied for these two examples.

- If $0 < \alpha_i < C$

$$b = b_1 = b - E_i - y_i(\alpha_i^* - \alpha_i^{(old)})x_i \cdot x_i - y_j(\alpha_j^* - \alpha_j^{(old)})x_i \cdot x_j \quad (17)$$

Step 4: Optimize the threshold b

We need the KKT conditions to be satisfied for these two examples.

- If $0 < \alpha_i < C$

$$b = b_1 = b - E_i - y_i(\alpha_i^* - \alpha_i^{(old)})x_i \cdot x_i - y_j(\alpha_j^* - \alpha_j^{(old)})x_i \cdot x_j \quad (17)$$

- If $0 < \alpha_j < C$

$$b = b_2 = b - E_j - y_i(\alpha_i^* - \alpha_i^{(old)})x_i \cdot x_j - y_j(\alpha_j^* - \alpha_j^{(old)})x_j \cdot x_j \quad (18)$$

Step 4: Optimize the threshold b

We need the KKT conditions to be satisfied for these two examples.

- If $0 < \alpha_i < C$

$$b = b_1 = b - E_i - y_i(\alpha_i^* - \alpha_i^{(old)})x_i \cdot x_i - y_j(\alpha_j^* - \alpha_j^{(old)})x_i \cdot x_j \quad (17)$$

- If $0 < \alpha_j < C$

$$b = b_2 = b - E_j - y_i(\alpha_i^* - \alpha_i^{(old)})x_i \cdot x_j - y_j(\alpha_j^* - \alpha_j^{(old)})x_j \cdot x_j \quad (18)$$

- If both α_i and α_j are at the bounds, then anything between b_1 and b_2 works, so we set

$$b = \frac{b_1 + b_2}{2} \quad (19)$$

Iterations / Details

- What if i doesn't violate the KKT conditions?
- What if $\eta \geq 0$?
- When do we stop?

Iterations / Details

- What if i doesn't violate the KKT conditions? **Skip it!**
- What if $\eta \geq 0$?
- When do we stop?

Iterations / Details

- What if i doesn't violate the KKT conditions? **Skip it!**
- What if $\eta \geq 0$? **Skip it!**
- When do we stop?

Iterations / Details

- What if i doesn't violate the KKT conditions? **Skip it!**
- What if $\eta \geq 0$? **Skip it!**
- When do we stop? **Until we go through α's without changing anything**

Plan

Dual Objective

Algorithm Big Picture

The Algorithm

Recap

Recap

- SMO: Optimize objective function for two data points
- Convex problem: Will converge
- Relatively fast
- Gives good performance
- Next HW!