Slack SVMs

Jordan Boyd-Graber
University of Colorado Boulder
LECTURE 8A
Can SVMs Work Here?

\[ y_i (w \cdot x_i + b) \geq 1 \]
Can SVMs Work Here?

\[ y_i(w \cdot x_i + b) \geq 1 \]  

(1)
Trick: Allow for a few bad apples
New objective function

\[
\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{\xi_i^p}
\]

subject to \(y_i(w \cdot x_i + b) \geq 1 - \xi_i \land \xi_i \geq 0, i \in [1, m]\)
New objective function

\[
\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i \quad (2)
\]

subject to \( y_i(w \cdot x_i + b) \geq 1 - \xi_i \wedge \xi_i \geq 0, i \in [1, m] \)

- Standard margin
New objective function

\[ \min_{w, b, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{\infty} \xi_i^p \]  

subject to \( y_i(w \cdot x_i + b) \geq 1 - \xi_i \wedge \xi_i \geq 0, i \in [1, m] \)

- Standard margin
- How wrong a point is (slack variables)
New objective function

\[
\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{m} \xi_i^{p}
\]

subject to \( y_i(w \cdot x_i + b) \geq 1 - \xi_i \land \xi_i \geq 0, i \in [1, m] \)

- Standard margin
- How wrong a point is (slack variables)
- Tradeoff between margin and slack variables
New objective function

\[
\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i^p
\]

subject to \( y_i(w \cdot x_i + b) \geq 1 - \xi_i \land \xi_i \geq 0, i \in [1, m] \)

- Standard margin
- How wrong a point is (slack variables)
- Tradeoff between margin and slack variables
- How bad wrongness scales
Aside: Loss Functions

- Losses measure how bad a mistake is
- Important for slack as well
Aside: Loss Functions

- Losses measure how bad a mistake is
- Important for slack as well
Aside: Loss Functions

- Losses measure how bad a mistake is
- Important for slack as well
Aside: Loss Functions

- Losses measure how bad a mistake is
- Important for slack as well

![Loss Functions Diagram]
Aside: Loss Functions

- Losses measure how bad a mistake is
- Important for slack as well

We’ll focus on linear hinge loss
Optimizing Constrained Functions

**Theorem: Lagrange Multiplier Method**

Given functions \( f(x_1, \ldots, x_n) \) and \( g(x_1, \ldots, x_n) \), the critical points of \( f \) restricted to the set \( g = 0 \) are solutions to equations:

\[
\frac{\partial f}{\partial x_i}(x_1, \ldots, x_n) = \lambda \frac{\partial g}{\partial x_i}(x_1, \ldots, x_n) \quad \forall i
\]

\( g(x_1, \ldots, x_n) = 0 \)

This is \( n + 1 \) equations in the \( n + 1 \) variables \( x_1, \ldots, x_n, \lambda \).
Lagrange Example

Maximize $f(x, y) = \sqrt{xy}$ subject to the constraint $20x + 10y = 200$.

- Compute derivatives

\[
\begin{align*}
\frac{\partial f}{\partial x} &= \frac{1}{2} \frac{\sqrt{y}}{x} \\
\frac{\partial g}{\partial x} &= 20 \\
\frac{\partial f}{\partial y} &= \frac{1}{2} \frac{\sqrt{x}}{y} \\
\frac{\partial g}{\partial y} &= 10
\end{align*}
\]
Lagrange Example

Maximize \( f(x, y) = \sqrt{xy} \) subject to the constraint \( 20x + 10y = 200 \).

- Compute derivatives

\[
\begin{align*}
\frac{\partial f}{\partial x} &= \frac{1}{2} \sqrt{\frac{y}{x}} & \frac{\partial g}{\partial x} &= 20 \\
\frac{\partial f}{\partial y} &= \frac{1}{2} \sqrt{\frac{x}{y}} & \frac{\partial g}{\partial y} &= 10
\end{align*}
\]
Lagrange Example

Maximize \( f(x, y) = \sqrt{xy} \) subject to the constraint \( 20x + 10y = 200 \).

• Compute derivatives

\[
\begin{align*}
\frac{\partial f}{\partial x} &= \frac{1}{2} \sqrt{\frac{y}{x}} \\
\frac{\partial g}{\partial x} &= 20 \\
\frac{\partial f}{\partial y} &= \frac{1}{2} \sqrt{\frac{x}{y}} \\
\frac{\partial g}{\partial y} &= 10
\end{align*}
\]

• Create new systems of equations
Lagrange Example

Maximize $f(x, y) = \sqrt{xy}$ subject to the constraint $20x + 10y = 200$.

- Compute derivatives
  
  \[
  \frac{\partial f}{\partial x} = \frac{1}{2} \sqrt{\frac{y}{x}} \quad \frac{\partial g}{\partial x} = 20 \\
  \frac{\partial f}{\partial y} = \frac{1}{2} \sqrt{\frac{x}{y}} \quad \frac{\partial g}{\partial y} = 10
  \]

- Create new systems of equations
  
  \[
  \frac{1}{2} \sqrt{\frac{y}{x}} = 20\lambda \\
  \frac{1}{2} \sqrt{\frac{x}{y}} = 10\lambda \\
  20x + 10y = 200
  \]
Lagrange Example

- Dividing the first equation by the second gives us
  \[
  \frac{y}{x} = 2 \tag{3}
  \]

- which means \( y = 2x \), plugging this into the constraint equation gives:
  \[
  20x + 20(2x) = 200 \\
  x = 5 \Rightarrow y = 10
  \]
New Lagrangian

\[ \mathcal{L}(\vec{w}, b, \xi, \alpha, \beta) = \frac{1}{2} ||w||^2 + C \sum_{i=1}^{m} \xi_i \] (4)

\[ - \sum_{i=1}^{m} \alpha_i \left[ y_i (w \cdot x_i + b) - 1 + \xi_i \right] \] (5)

\[ - \sum_{i=1}^{m} \beta_i \xi_i \] (6)
New Lagrangian

\[ \mathcal{L}(\tilde{w}, b, \tilde{\xi}, \tilde{\alpha}, \tilde{\beta}) = \frac{1}{2} ||w||^2 + C \sum_{i=1}^{m} \xi_i \]  

\[ - \sum_{i=1}^{m} \alpha_i [y_i(w \cdot x_i + b) - 1 + \xi_i] \]  

\[ - \sum_{i=1}^{m} \beta_i \xi_i \]

Taking the gradients \( \nabla_w \mathcal{L}, \nabla_b \mathcal{L}, \nabla_{\xi_i} \) and solving for zero gives us

\[ \sum_{i=1}^{m} \alpha_i y_i = 0 \]  

\[ \tilde{w} = \sum_{i=1}^{m} \alpha_i y_i x_i \]  

\[ \alpha_i + \beta_i = C \]
New Lagrangian

\[
\mathcal{L}(\vec{w}, b, \xi, \alpha, \beta) = \frac{1}{2} ||w||^2 + C \sum_{i=1}^{m} \xi_i \tag{4}
\]

\[
- \sum_{i=1}^{m} \alpha_i [y_i(w \cdot x_i + b) - 1 + \xi_i] \tag{5}
\]

\[
- \sum_{i=1}^{m} \beta_i \xi_i \tag{6}
\]

Taking the gradients (\(\nabla_w \mathcal{L}, \nabla_b \mathcal{L}, \nabla_{\xi_i}\)) and solving for zero gives us

\[
\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{7}
\]

\[
\vec{w} = \sum_{i=1}^{m} \alpha_i y_i x_i \tag{8}
\]

\[
\alpha_i + \beta_i = C \tag{9}
\]
New Lagrangian

\[ \mathcal{L}(\tilde{w}, b, \tilde{\xi}, \tilde{\alpha}, \tilde{\beta}) = \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i \] (4)

\[ - \sum_{i=1}^{m} \alpha_i [y_i(w \cdot x_i + b) - 1 + \xi_i] \] (5)

\[ - \sum_{i=1}^{m} \beta_i \xi_i \] (6)

Taking the gradients (\(\nabla_w \mathcal{L}, \nabla_b \mathcal{L}, \nabla_{\xi_i}\)) and solving for zero gives us

\[ \sum_{i=1}^{m} \alpha_i y_i = 0 \] (7)

\[ \tilde{w} = \sum_{i=1}^{m} \alpha_i y_i x_i \] (8)

\[ \alpha_i + \beta_i = C \] (9)
New Lagrangian

\[ \mathcal{L}(\mathbf{\tilde{w}}, b, \xi, \tilde{\alpha}, \tilde{\beta}) = \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_{i=1}^{m} \xi_i \] (4)

\[- \sum_{i=1}^{m} \alpha_i \left[ y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i \right] \] (5)

\[- \sum_{i=1}^{m} \beta_i \xi_i \] (6)

Taking the gradients (\( \nabla_w \mathcal{L}, \nabla_b \mathcal{L}, \nabla_{\xi_i} \)) and solving for zero gives us

\[ \sum_{i=1}^{m} \alpha_i y_i = 0 \] (7)

\[ \mathbf{\tilde{w}} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \] (8)

\[ \alpha_i + \beta_i = C \] (9)
Simplifying dual objective

\[
\sum_{i=1}^{m} \alpha_i y_i = 0 \\
\vec{w} = \sum_{i=1}^{m} \alpha_i y_i x_i \\
\alpha_i + \beta_i = C
\]
Simplifying dual objective

\[ \sum_{i=1}^{m} \alpha_i y_i = 0 \]

\[ \vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \]

\[ \alpha_i + \beta_i = C \]

\[ \mathcal{L} = \frac{1}{2} \| \vec{w} \| - \sum_{i} \alpha_i y_i \vec{w} \cdot \vec{x}_i - \sum_{i} \alpha_i y_i b - \sum \beta_i \xi_i \quad (10) \]
Simplifying dual objective

\[ \sum_{i=1}^{m} \alpha_i y_i = 0 \quad \text{\textbf{\(\hat{w} = \sum_{i=1}^{m} \alpha_i y_i x_i\)}} \quad \alpha_i + \beta_i = C \]

\[ \mathcal{L} = \frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i \bar{x}_i \right\| - \sum_{i}^{m} \sum_{j}^{m} \alpha_i \alpha_j y_i y_j (\bar{x}_j \cdot \bar{x}_i) - \sum_{i}^{m} \alpha_i y_i b - \sum_{i=1}^{m} \beta_i \xi_i \]

(10)
Simplifying dual objective

\[ \sum_{i=1}^{m} \alpha_i y_i = 0 \quad \text{and} \quad \vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \quad \text{such that} \quad \alpha_i + \beta_i = C \]

\[ \mathcal{L} = \frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \right\| - \sum_{i} \sum_{j} \alpha_i \alpha_j y_i y_j (\vec{x}_j \cdot \vec{x}_i) - \sum_{i} \alpha_i y_i b - \sum_{i=1}^{m} \beta_i \xi_i \]

(10)
Simplifying dual objective

\[ \sum_{i=1}^{m} \alpha_i y_i = 0 \quad \bowtie \quad \bar{w} = \sum_{i=1}^{m} \alpha_i y_i x_i \quad \alpha_i + \beta_i = C \]

\[ \mathcal{L} = \frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i \tilde{x}_i \right\| - \sum_{i} \sum_{j} \alpha_i \alpha_j y_i y_j (\tilde{x}_j \cdot \tilde{x}_i) - \sum_{i} \alpha_i y_i b + \sum_{i} \alpha_i \]

(10)
Simplifying dual objective

\[ \sum_{i=1}^{m} \alpha_i y_i = 0 \]

\[ \vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \]

\[ \alpha_i + \beta_i = C \]

\[ \mathcal{L} = \frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \right\| - \sum_{i} \sum_{j} \alpha_i \alpha_j y_i y_j (\vec{x}_j \cdot \vec{x}_i) - \sum_{i} \alpha_i y_i b + \sum_{i} \alpha_i \]

(10)
Simplifying dual objective

\[ \sum_{i=1}^{m} \alpha_i y_i = 0 \]

\[ \vec{w} = \sum_{i=1}^{m} \alpha_i y_i x_i \]

\[ \alpha_i + \beta_i = C \]

\[ L = \frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \right\| - \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\vec{x}_j \cdot \vec{x}_i) - \sum_{i=1}^{m} \alpha_i y_i b + \sum_{i=1}^{m} \alpha_i \]

(10)
Simplifying dual objective

\[
\sum_{i=1}^{m} \alpha_i y_i = 0 \\
\tilde{w} = \sum_{i=1}^{m} \alpha_i y_i x_i \\
\alpha_i + \beta_i = C
\]

\[
\mathcal{L} = \frac{1}{2} \left\| \sum_{i=1}^{m} \alpha_i y_i \tilde{x}_i \right\| - \sum_{i} \sum_{j} \alpha_i \alpha_j y_i y_j (\tilde{x}_j \cdot \tilde{x}_i) + \sum_{i} \alpha_i
\]  \hspace{1cm} (10)

First two terms are the same!
Simplifying dual objective

\[ \sum_{i=1}^{m} \alpha_i y_i = 0 \]
\[ \vec{w} = \sum_{i=1}^{m} \alpha_i y_i x_i \]
\[ \alpha_i + \beta_i = C \]

\[ \mathcal{L} = -\frac{1}{2} \sum_{i}^{m} \sum_{j}^{m} \alpha_i \alpha_j y_i y_j (\vec{x}_j \cdot \vec{x}_i) + \sum_{i}^{m} \alpha_i \]

(10)

Just like separable case, except that we add the constraint that \( \alpha_i \leq C \)!
Wrapup

- Adding slack variables don’t break the SVM problem
- Very popular algorithm
  - SVMLight (many options)
  - Libsvm / Liblinear (very fast)
  - Weka (friendly)
  - pyml (Python focused, from Colorado)
- Next up: simple algorithm for finding SVMs
Plan

Dual Objective

Algorithm Big Picture

The Algorithm

Recap
Lagrange Multipliers

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

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i [y_i(w \cdot x_i + b) - 1]$$  (11)
Lagrange Multipliers

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

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{m} \alpha_i [y_i(w \cdot x_i + b) - 1] \quad (11)$$

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

Weights

\[ \vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \]  

(12)
Solving Lagrangian

Weights

\[ \vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \]  

(12)

Bias

\[ 0 = \sum_{i=1}^{m} \alpha_i y_i \]  

(13)
The document contains the following content:

**Dual Objective**

### Solving Lagrangian

#### Weights

\[ \vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i \]  

(12)

#### Bias

\[ 0 = \sum_{i=1}^{m} \alpha_i y_i \]  

(13)

#### Support Vector-ness

\[ \alpha_i = 0 \lor y_i (w \cdot x_i + b) = 1 \]  

(14)
Reparameterize in terms of $\alpha$

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

(15)
Outline for SVM Optimization (SMO)

1. Select two examples \( i, j \)
2. Get a learning rate \( \eta \)
3. Update \( \alpha_j \)
4. Update \( \alpha_i \)
Plan

Dual Objective

Algorithm Big Picture

The Algorithm

Recap
Contrast with SG

- There’s a learning rate $\eta$ that depends on the data
- Use the error of an example to derive update
- You update multiple $\alpha$ at once
Contrast with SG

- There’s a learning rate $\eta$ that depends on the data
- Use the error of an example to derive update
- You update multiple $\alpha$ 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?
  
  $\alpha_i = 0 \Rightarrow y_i(w \cdot x_i + b) \geq 1$  
  (16)

  $\alpha_i = C \Rightarrow y_i(w \cdot x_i + b) \leq 1$  
  (17)

  $0 < \alpha_i < C \Rightarrow y_i(w \cdot x_i + b) = 1$  
  (18)

(Karush-Kuhn-Tucker Conditions)
More details

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

$$\alpha_i = 0 \Rightarrow y_i (w \cdot x_i + b) \geq 1$$  \hspace{1cm} (16)

$$\alpha_i = C \Rightarrow y_i (w \cdot x_i + b) \leq 1$$  \hspace{1cm} (17)

$$0 < \alpha_i < C \Rightarrow y_i (w \cdot x_i + b) = 1$$  \hspace{1cm} (18)

(Karush-Kuhn-Tucker Conditions)

- Keep checking (to some tolerance)
Plan

Dual Objective

Algorithm Big Picture

The Algorithm

Recap
Step 1: Select $i$ and $j$

- Iterate over $i = \{1, \ldots, m\}$
- 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 $\alpha_j$

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

For $y_i \neq y_j$:

- $y_i = y_j$
  
  $L = \max(0, \alpha_j - \alpha_i)$ \hspace{1cm} (19)
  
  $H = \min(C, C + \alpha_j - \alpha_i)$ \hspace{1cm} (20)

For $y_i = y_j$:

  $L = \max(0, \alpha_i + \alpha_j - C)$ \hspace{1cm} (21)
  
  $H = \min(C, \alpha_j + \alpha_i)$ \hspace{1cm} (22)
Step 2: Optimize $\alpha_j$

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

$y_i \neq y_j$

- $L = \max(0, \alpha_j - \alpha_i)$ \hspace{1cm} (19)
- $H = \min(C, C + \alpha_j - \alpha_i)$ \hspace{1cm} (20)

$y_i = y_j$

- $L = \max(0, \alpha_i + \alpha_j - C)$ \hspace{1cm} (21)
- $H = \min(C, \alpha_j + \alpha_i)$ \hspace{1cm} (22)

This is because the update for $\alpha_i$ is based on $y_i y_j$ (sign matters).
Step 2: Optimize $\alpha_j$

Compute errors for $i$ and $j$

$$E_k \equiv f(x_k) - y_k$$  \hspace{1cm} (23)
Step 2: Optimize $\alpha_j$

Compute errors for $i$ and $j$

$$E_k \equiv f(x_k) - y_k$$  \hspace{1cm} (23)

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

$$\eta = 2x_i \cdot x_j - x_i \cdot x_i - x_j \cdot x_j$$  \hspace{1cm} (24)
**Step 2: Optimize $\alpha_j$**

Compute errors for $i$ and $j$

$$E_k \equiv f(x_k) - y_k \tag{23}$$

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

$$\eta = 2x_i \cdot x_j - x_i \cdot x_i - x_j \cdot x_j \tag{24}$$

for new value for $\alpha_j$

$$\alpha_j^* = \alpha_j^{(old)} - \frac{y_j(E_i - E_j)}{\eta} \tag{25}$$

Similar to stochastic gradient, but with additional error term.

If $\alpha_j^*$ is outside $[L, H]$, clip it so that it is within the range.
Step 2: Optimize $\alpha_j$

Compute errors for $i$ and $j$

$$E_k \equiv f(x_k) - y_k$$ \hfill (23)

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

$$\eta = 2x_i \cdot x_j - x_i \cdot x_i - x_j \cdot x_j$$ \hfill (24)

for new value for $\alpha_j$

$$\alpha_j^* = \alpha_j^{(old)} - \frac{y_j(E_i - E_j)}{\eta}$$ \hfill (25)

Similar to stochastic gradient, but with additional error term.
Step 2: Optimize \( \alpha_j \)

Compute errors for \( i \) and \( j \)

\[
E_k \equiv f(x_k) - y_k
\]

(23)

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

\[
\eta = 2x_i \cdot x_j - x_i \cdot x_i - x_j \cdot x_j
\]

(24)

for new value for \( \alpha_j \)

\[
\alpha_j^* = \alpha_j^{(old)} - \frac{y_j(E_i - E_j)}{\eta}
\]

(25)

Similar to stochastic gradient, but with additional error term. If \( \alpha_j^* \) is outside \([L, H]\), clip it so that it is within the range.
Step 3: Optimize $\alpha_i$

Set $\alpha_i$:

$$\alpha_i^* = \alpha_i^{(old)} + y_i y_j \left( \alpha_j^{(old)} - \alpha_j \right)$$  \hspace{1cm} (26)
Step 3: Optimize $\alpha_i$

Set $\alpha_i$:

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

This balances out the move that we made for $\alpha_j$. 

(26)
Step 4: Optimize the threshold $b$

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

- **If $0 < \alpha_i < C$**

$$b = b_1 = b - E_i - y_i(\alpha_i^* - \alpha_i^{(old)})x_i \cdot x_i - y_j(\alpha_j^* - \alpha_j^{(old)})x_i \cdot x_j$$ (27)
Step 4: Optimize the threshold $b$

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

- If $0 < \alpha_i < C$

$$b = b_1 = b - E_i - y_i(\alpha_i^* - \alpha_i^{(old)})x_i \cdot x_i - y_j(\alpha_j^* - \alpha_j^{(old)})x_i \cdot x_j \quad (27)$$

- If $0 < \alpha_j < C$

$$b = b_2 = b - E_j - y_i(\alpha_i^* - \alpha_i^{(old)})x_i \cdot x_j - y_j(\alpha_j^* - \alpha_j^{(old)})x_j \cdot x_j \quad (28)$$
Step 4: Optimize the threshold $b$

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

- If $0 < \alpha_i < C$

  $$b = b_1 = b - E_i - y_i(\alpha_i^* - \alpha_i^{(old)})x_i \cdot x_i - y_j(\alpha_j^* - \alpha_j^{(old)})x_i \cdot x_j$$  (27)

- If $0 < \alpha_j < C$

  $$b = b_2 = b - E_j - y_i(\alpha_i^* - \alpha_i^{(old)})x_i \cdot x_j - y_j(\alpha_j^* - \alpha_j^{(old)})x_j \cdot x_j$$  (28)

- If both $\alpha_i$ and $\alpha_j$ are at the bounds, then anything between $b_1$ and $b_2$ works, so we set

  $$b = \frac{b_1 + b_2}{2}$$  (29)
• What if \( i \) doesn’t violate the KKT conditions?
• What if \( \eta \geq 0? \)
• When do we stop?
The Algorithm

Iterations / Details

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

Iterations / Details

- What if \( i \) doesn’t violate the KKT conditions? **Skip it!**
- What if \( \eta \geq 0? \) **Skip it!**
- When do we stop?
The Algorithm

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 $\alpha$’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!