Optimization Problem

Recall the following setting for training support vector machines.

Assume that we are given the training set,

$$D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subseteq \mathbb{R}^n \times \{+1, -1\}.$$

We are interested in computing a classifier in the form of a support vector machine model,

$$\hat{f}(\overline{x}) = \text{sign}\left(\sum_{i=1}^{l} y_i \alpha_i^* k(\overline{x}_i, \overline{x}) - b^*\right),$$

using a training algorithm based on the Lagrangian dual,

$$\overline{\alpha}^* = \underset{\overline{\alpha}}{\text{argmax}}\, \phi'(\overline{\alpha}) = \underset{\overline{\alpha}}{\text{argmax}} \left(\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(\overline{x}_i, \overline{x}_j)\right),$$

subject to the constraints,

$$\sum_{i=1}^{l} y_i \alpha_i = 0,$$
$$C \geq \alpha_i \geq 0,$$

with $i = 1, \ldots, l$.

Quadratic Programming

We can use quadratic programming to implement support vector machines,

$$\overline{\alpha}^* = \underset{\overline{\alpha}}{\operatorname{argmin}} \left(\frac{1}{2} \overline{\alpha}^T \mathbf{Q} \overline{\alpha} - \overline{q} \bullet \overline{\alpha} \right),$$

subject to the constraints,

$$\mathbf{Y}\overline{\alpha} = 0,$$

$$\overline{u} \leq \overline{\alpha} \leq \overline{v}.$$

In order to do this we have to use quadratic programming packages that support both equality and inequality constraints.

This is different from our use of quadratic programming in the primal maximum margin classifiers - there we only needed to support inequality constraints,

$$\overline{w}^* = \underset{\overline{w}}{\operatorname{argmin}} \left(\frac{1}{2} \overline{w}^T \mathbf{Q}\, \overline{w} - \overline{q} \bullet \overline{w} \right),$$

subject to the constraints

$$\mathbf{X}^T \overline{w} \geq \overline{c}.$$

Quadratic Programming

Quadratic solvers are usually defined in terms of minimization, therefore we need to massage our support vector optimization problem a little bit,

$$\begin{aligned}\underset{\overline{\alpha}}{\operatorname{argmax}}\, \phi'(\overline{\alpha}) &= \underset{\overline{\alpha}}{\operatorname{argmin}}\, \left(-\phi'(\overline{\alpha})\right) \\ &= \underset{\overline{\alpha}}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(\overline{x}_i, \overline{x}_j) - \sum_{i=1}^{l} \alpha_i \right) \\ &= \underset{\overline{\alpha}}{\operatorname{argmin}} \left(\frac{1}{2} \overline{\alpha}^T \mathbf{Q} \overline{\alpha} - \overline{q} \bullet \overline{\alpha} \right)\end{aligned}$$

where the matrix $\mathbf{Q}$ is an $l \times l$ matrix with components

$$Q_{ij} = y_i y_j k(\overline{x}_i, \overline{x}_j)$$

and the vector $\overline{q}$ has l components initialized to one,

$$\overline{q} = \overline{1}.$$

We call the matrix $\mathbf{Q}$ the *kernel matrix*. In effect, when using quadratic programming solvers we need to *precompute* the dot product between the pairwise training points.

Quadratic Programming

The constraints

$$\mathbf{Y}\overline{\alpha} = 0,$$
$$\overline{u} \leq \overline{\alpha} \leq \overline{v}.$$

are easily instantiated. We let $\mathbf{Y}$ be a $1 \times l$ matrix with components

$$Y_{1i} = y_i,$$

where $(\overline{x}_i, y_i) \in D$ then,

$$\mathbf{Y}\overline{\alpha} = \sum_{i=1}^{l} y_i \alpha_i.$$

Finally, if we let $\overline{u} = \overline{0}$ and let vector $\overline{v}$ be a constant vector with $v_i = C$ then,

$$u_i \leq \alpha_i \leq v_i,$$

which implies

$$0 \leq \alpha_i \leq C,$$

with $i = 1, \ldots, l$.

Quadratic Programming

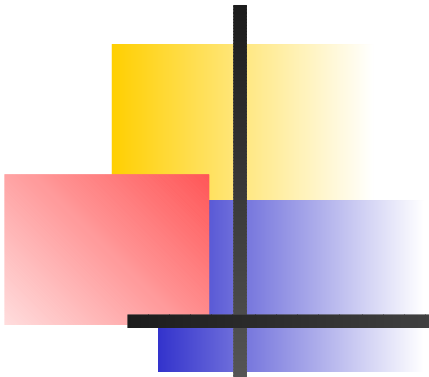
let $D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subset \mathbb{R}^n \times \{+1, -1\}$
let $C > 0$
let $\mathbf{Q}$ be a $l \times l$ matrix with components $Q_{ij} = y_i y_j k(\overline{x}_i, \overline{x}_j)$
let $\mathbf{Y}$ be a $1 \times l$ matrix with components $Y_{1j} = y_j$
let $\overline{q}$ be a constant vector with $q_i = 1$
let $\overline{u}$ be a constant vector with $u_i = 0$
let $\overline{v}$ be a constant vector with $v_i = C$
$\overline{\alpha} \leftarrow \mathsf{solve}(\mathbf{Q}, \overline{q}, \mathbf{Y}, \overline{u}, \overline{v})$
$b \leftarrow$ compute using known support vectors
return $(\overline{\alpha}, b)$

Using a quadratic programming solver to train support vector machines.

Quadratic Programming

Observation: Our kernel matrix $\mathbf{Q}$ is an $l \times l$ matrix on the training set D. Consider a large training set with 50,000 observations. This would mean that the kernel matrix contains 2.5 billion elements, or consumes 10GB of memory if we represent each element with a 4 byte word.

$\Rightarrow$ Not feasible on most of today's architectures.



Chunking

Idea: For most data sets the ratio of support vectors to training instances is very low; we say that solving the dual optimization problem for SVMs gives rise to a *sparse solution*.

The idea is then to break up the training data into *chunks* and search for support vectors in the individual chunks.

If we keep the size of the chunks or *working sets* small then the original optimization problem breaks down into smaller, manageable optimization sub-problems.

Chunking

let $D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subset \mathbb{R}^n \times \{+1, -1\}$
let $k > 0$
$\overline{\alpha} \leftarrow \overline{0}$
select a subset W of size k from D
repeat forever
  solve Lagrangian dual for W (update $\overline{\alpha}$ accordingly)
  delete observations from W that are not support vectors
  $b \leftarrow$ compute using support vectors in W
  if all $d \in D$ satisfy the KKT conditions **then**
    return $(\overline{\alpha}, b)$
  end if
  $D_k \leftarrow$ the k worst offenders in D of the KKT conditions
  $W \leftarrow W \cup D_k$
end repeat

Solving the dual optimization problem using chunking.

Observation: $|W \cup D_k| \approx k$ when the solution is sparse.

Chunking

We are trying to find a global maximum by solving smaller sub-problems - how do we know when we are done?

$\Rightarrow$ We monitor the KKT conditions,

$$\begin{aligned}
\frac{\partial L}{\partial \overline{w}}(\overline{\alpha}, \overline{\beta}, \overline{w}^*, \overline{\xi}, b) &= 0, \\
\frac{\partial L}{\partial \xi_i}(\overline{\alpha}, \overline{\beta}, \overline{w}, \xi_i^*, b) &= 0, \\
\frac{\partial L}{\partial b}(\overline{\alpha}, \overline{\beta}, \overline{w}, \overline{\xi}, b^*) &= 0, \\
\alpha_i^*(y_i(\overline{w}^* \bullet \overline{x}_i - b^*) + \xi_i^* - 1) &= 0, \\
\beta_i^* \xi_i^* &= 0, \\
y_i(\overline{w}^* \bullet \overline{x}_i - b^*) + \xi_i^* - 1 &\geq 0, \\
\alpha_i^* &\geq 0, \\
\beta_i^* &\geq 0, \\
\xi_i^* &\geq 0,
\end{aligned}$$

for $i = 1, \ldots, l$.

Chunking

A closer look reveals that we only need to monitor the complimentarity conditions,

$$\alpha_i(y_i(\overline{w} \bullet \overline{x}_i - b) + \xi_i - 1) = 0,$$
$$\beta_i \xi_i = 0.$$

We can rewrite these slightly using the dual for the normal vector $\overline{w}$ and $\beta_i = C - \alpha_i$ as,

$$\alpha_i \left(y_i \left(\sum_{j=1}^{l} y_j \alpha_j k(\overline{x}_j, \overline{x}_i) - b \right) - 1 + \xi_i \right) = 0,$$
$$(C - \alpha_i)\xi_i = 0,$$

for all $(\overline{x}_i, y_i) \in D$ with $i = 1, \ldots, l$.

Problem: We cannot directly monitor the slack variables ξ_i. However, we can obtain a set of conditions implied by the complimentarity conditions using a case analysis on α_i.

Chunking

Case analysis:

$\alpha_i = 0$**:** This implies $\xi_i = 0$, therefore $y_i \left(\sum_{j=1}^{l} y_j \alpha_j k(\overline{x}_j, \overline{x}_i) - b \right) \geq 1.$

$0 < \alpha_i < C$**:** This implies $\xi_i = 0$, therefore $y_i \left(\sum_{j=1}^{l} y_j \alpha_j k(\overline{x}_j, \overline{x}_i) - b \right) = 1.$

$\alpha_i = C$**:** This implies $\xi_i > 0$, therefore $y_i \left(\sum_{j=1}^{l} y_j \alpha_j k(\overline{x}_j, \overline{x}_i) - b \right) \leq 1.$

Chunking

Putting it all together gives us the condition:

$$y_i \left(\sum_{j=1}^{l} y_j \alpha_j k(\overline{x}_j, \overline{x}_i) - b \right) \begin{cases} \geq 1 \text{ if } \alpha_i = 0 \\ = 1 \text{ if } 0 < \alpha_i < C \\ \leq 1 \text{ if } \alpha_i = C \end{cases}$$

for all $(\overline{x}_i, y_i) \in D$ with $i = 1, \ldots, l$.

Observation: One way to view these conditions is as a *consistency test*: if the values of the Lagrangian multipliers are consistent with the locations of the corresponding training points vis-à-vis a decision surface, then we have found a global solution.

SMO

SMO – Sequential Minimal Optimization.

We can view the SMO algorithm as a chunking algorithm with a working set of size 2.

A working set of size 2 is the smallest working set that allows for an optimization that satisfies the constraints,

$$\sum_{i=1}^{l} y_i \alpha_i = 0.$$

Then consider the subproblem of our Lagrangian dual optimization using the Lagrangian multipliers α_j and α_k (our working set),

$$\max_{\alpha_j, \alpha_k} \phi'(\alpha_j, \alpha_k),$$

subject to the constraints,

$$y_j \alpha_j + y_k \alpha_k = \delta,$$
$$C \geq \alpha_j, \alpha_k \geq 0,$$

with $j, k = 1, \ldots, l$ and $j \neq k$.

SMO

We can rewrite the second to last equation on the previous slide as,

$$\alpha_k = y_k(\delta - y_j\alpha_j).$$

It follows that,

$$\max_{\alpha_j,\alpha_k} \phi'(\alpha_j, \alpha_k) = \max_{\alpha_j} \phi'\left(\alpha_j, y_k(\delta - y_j\alpha_j)\right) = \max_{\alpha_j} \psi(\alpha_j).$$

That is, our two-variable optimization problem becomes an optimization problem over the single variable α_j. We represent the objective function of this single variable optimization problem as $\psi(\alpha_j)$. Now ψ has a single maximum that we can find by,

$$\frac{d\psi}{d\alpha_j} = 0,$$

and solving for α_j and then use above equation to fiind α_k.

Some care needs to be taken to satisfy the constraints on α_i during this optimization.

SMO

let $D = \{(\overline{x}_1, y_1), (\overline{x}_2, y_2), \ldots, (\overline{x}_l, y_l)\} \subset \mathbb{R}^n \times \{+1, -1\}$
$\overline{\alpha} \leftarrow \overline{0}$

repeat

1. pick two points, $\overline{x}_j$ and $\overline{x}_k$ in D together with their respective Lagrangian multipliers, α_j and α_k, where $j \neq k$.
2. optimize the subproblem $\max \phi'(\alpha_j, \alpha_k)$ (keeping the other Lagrangian multipliers constant).
3. compute b

until the KKT conditions hold for all $d \in D$

return $(\overline{\alpha}, b)$

Sequential Minimal Optimization.