Gradient Ascent

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 $i = 1, \ldots, l$.

Gradient Ascent

In order to train SVMs we need to solve the optimization problem

$$\overline{\alpha}^* = \underset{\overline{\alpha}}{\operatorname{argmax}}\, \phi'(\overline{\alpha}).$$

Perhaps the most straightforward implementation of the Lagrangian dual optimization problem is by *gradient ascent*.

Formally, let h be a differentiable function with respect to $\overline{x} \in \mathbb{R}^n$, then the *gradient* of h is defined as,

$$\nabla h = \left(\frac{\partial h}{\partial x_1}, \cdots, \frac{\partial h}{\partial x_n} \right).$$

We often write,

$$\nabla_i h = \frac{\partial h}{\partial x_i},$$

for the i^{th} component of ∇h with $i = 1, \ldots, n$.

Now, $\nabla h(\overline{y})$ with $\overline{y} \in \mathbb{R}^n$ is a vector that points in the direction of the largest increase of h at point $\overline{y}$.

We can use this to find the maximum of our dual ϕ' by simply following the gradient until the gradient becomes zero $\Rightarrow$ gradient ascent.

Gradient Ascent

let $\eta \in [0, 1]$
$\overline{\alpha} \leftarrow \overline{0}$
repeat
  $\overline{\alpha}_{\text{old}} \leftarrow \overline{\alpha}$
  for $i = 1$ **to** l **do**
    $\alpha_i \leftarrow \alpha_i + \eta \nabla_i \phi'(\overline{\alpha})$
  end for
until $\overline{\alpha} - \overline{\alpha}_{\text{old}} \approx \overline{0}$
return $\overline{\alpha}$

The gradient ascent algorithm.

Gradient Ascent

Observation: We have treated our optimization problem as an unconstrained optimization problem ignoring the constraints,

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

with $i = 1, \ldots, l$. The first constraint is due to the optimization of the offset term b and the second constraint is the soft-margin constraint for the Lagrangian multipliers.

We can dispense with the first constraint by simply setting $b = 0$.

The second constraint is easily implemented as a set of *box constraints* on $\overline{\alpha}$,

$$\alpha_i \leftarrow \min\left\{C, \max\left\{0, \alpha_i + \eta \nabla_i \phi'(\overline{\alpha})\right\}\right\}.$$

The Kernel-Adatron

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 $\eta > 0$
let $C > 0$
let $b = 0$
$\overline{\alpha} \leftarrow \overline{0}$
repeat
  $\overline{\alpha}_{\text{old}} \leftarrow \overline{\alpha}$
  for $i = 1$ **to** l **do**
    $\alpha_i \leftarrow \min\left\{C, \max\left\{0, \alpha_i + \eta - \eta y_i \sum_{j=1}^{l} y_j \alpha_j k(\overline{x}_j, \overline{x}_i)\right\}\right\}$
  end for
until $\overline{\alpha} - \overline{\alpha}_{\text{old}} \approx \overline{0}$
return $(\overline{\alpha}, b)$

The Kernel-Adatron algorithm.