SVM by Sequential Minimal Optimization (SMO)

www.cs.wisc.edu/~dpage

Quick Review

As last lecture showed us, we can

- Solve the dual more efficiently (fewer unknowns)
- Add parameter C to allow some misclassifications
- Replace $\mathbf{x}_i^T\mathbf{x}_j$ by more more general kernel term

$$\min_{\vec{\alpha}} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j K(\vec{x}_i,\vec{x}_j)\alpha_i\alpha_j - \sum_{i=1}^{N}\alpha_i,$$
$$0 \le \alpha_i \le C, \forall i,$$
$$\sum_{i=1}^{N} y_i\alpha_i = 0.$$

Intuitive Introduction to SMO

- Perceptron learning algorithm is essentially doing same thing – find a linear separator by adjusting weights on misclassified examples
- Unlike perceptron, SMO has to maintain sum over examples of example weight times example label
- Therefore, when SMO adjusts weight of one example, it must also adjust weight of another

Intuitive Introduction to SMO

- Can view weight vector for perceptron as weighted sum of examples:

$$(w_1, w_2, \ldots, w_N) = \alpha_1 (x_1^1, x_2^1, \ldots, x_N^1) + \alpha_2 (x_1^2, x_2^2, \ldots, x_N^2) + \ldots + \alpha_M (x_1^M, x_2^M, \ldots, x_N^M)$$

- Learning: Repeat until convergence property met
  - Randomly choose an example
  - If mislabeled, add to its weight

w

- SMO does same, except must respect constraint:

$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0$$

Intuitive Intro (Continued)

- This constraint means whenever we add an amount β to an α_i, we have to either:
  - subtract β from the α_j for another example with the same sign (class label), or
  - add β to the α_j for another example with the opposite sign (class label)
- Therefore at each step we have to randomly choose **two** examples whose weights to revise, and size of revision depends on difference in their errors
- For soft margin SVM, we also have to "clip" size of any change because of additional constraint that every α must be between 0 and C

Recall Perceptron as Classifier

- Output for example **x** is sign($\mathbf{w}^T\mathbf{x}$)
- Candidate Hypotheses: real-valued weight vectors **w**
- Training: Update **w** for each misclassified example **x** (target class *y*, predicted *o*) by:
  - $w_i \leftarrow w_i + \eta(y\text{-}o)x_i$
  - η is learning rate parameter
- Let E be the error *o-y*. The above can be rewritten as

$$\mathbf{w} \leftarrow \mathbf{w} - \eta E \mathbf{x}$$

- So final weight vector is a sum of weighted contributions from each example. Predictive model can be rewritten as weighted combination of examples rather than features, where the weight on an example is sum (over iterations) of terms $-\eta E$.

Corresponding Use in Prediction

- To use perceptron, prediction is $\mathbf{w}^T\mathbf{x}$ - b. May treat -b as one more coefficient in w (and omit it here), may take sign of this value
- In alternative view of last slide, prediction can be written as $\sum_j a_j'(\mathbf{x}_j^T \mathbf{x}) - b$ or, revising the weight appropriately: $\sum_j y_j a_j(\mathbf{x}_i^T \mathbf{x}) - b$
- Prediction in SVM is: $u = \sum_{j=1}^{N} y_j \alpha_j K(\vec{x}_j, \vec{x}) - b$

From Perceptron Rule to SMO Rule

- Recall that SVM optimization problem has the added requirement that: $\sum_{i=1}^{N} y_i \alpha_i = 0$
- Therefore if we increase one α by an amount η, in either direction, then we have to change another α by an equal amount in the opposite direction (relative to class value). We can accomplish this by changing: $\mathbf{w} \leftarrow \mathbf{w} - \eta E \mathbf{x}$ also viewed as:

$$\alpha \leftarrow \alpha - y\eta E$$ to:

$$\alpha_1 \leftarrow \alpha_1 - y_1\eta(E_1 - E_2)$$

or equivalently: $\alpha_1 \leftarrow \alpha_1 + y_1\eta(E_2 - E_1)$

We then also adjust α_2 so that $y_1\alpha_1 + y_2\alpha_2$ stays same

First of Two Other Changes from Perceptron Rule

- Instead of learning rate of $\frac{1}{20}$ or some similar constant, we use $\frac{1}{x_1 \cdot x_1 + x_2 \cdot x_2 - 2x_1 \cdot x_2}$

  or more generally $\frac{1}{K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2)}$

- For a given difference in error between two examples, the step size is larger when the two examples are more similar. When x_1 and x_2 are similar, $K(x_1,x_2)$ is larger (for typical kernels, including dot product). Hence the denominator is smaller and step size is larger.

Second of Two Other Changes

- Recall our formulation had an additional constraint:

$$0 \leq \alpha_i \leq C, \forall i$$

- Therefore, we ¨clip´´ any change we make to any α to respect this constraint

- This yields the following SMO algorithm. (So far we have motivated it intuitively. Afterward we will show it really minimizes our objective.)

Updating Two α's: One SMO Step

- Given examples e1 and e2, set

$$\alpha_2^{\text{new}} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta}$$

where η = $K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2)$

- Clip this value in the natural way:

$$\alpha_2^{\text{new,clipped}} = \begin{cases} H & \text{if} \quad \alpha_2^{\text{new}} \geq H; \\ \alpha_2^{\text{new}} & \text{if} \quad L < \alpha_2^{\text{new}} < H; \\ \text{L} & \text{if} \quad \alpha_2^{\text{new}} \leq L. \end{cases}$$

if y1 = y2 then:

$$L = \max(0, \alpha_2 + \alpha_1 - C)$$
$$H = \min(C, \alpha_2 + \alpha_1)$$

otherwise:

$$L = \max(0, \alpha_2 - \alpha_1)$$
$$H = \min(C, C + \alpha_2 - \alpha_1)$$

- Set $\alpha_1^{\text{new}} = \alpha_1 + s(\alpha_2 - \alpha_2^{\text{new,clipped}})$ where $s = y_1 y_2$

Karush-Kuhn-Tucker (KKT) Conditions

- It is necessary and sufficient for a solution to our objective that all α's satisfy the following:

$$\alpha_i = 0 \Leftrightarrow y_i u_i \geq 1,$$
$$0 < \alpha_i < C \Leftrightarrow y_i u_i = 1,$$
$$\alpha_i = C \Leftrightarrow y_i u_i \leq 1.$$

- An α is 0 iff that example is correctly labeled with room to spare
- An α is C iff that example is incorrectly labeled or in the margin
- An α is properly between 0 and C (is "unbound") iff that example is "barely" correctly labeled (is a support vector)

- We just check KKT to within some small epsilon, typically 10^{-3}

SMO Algorithm

- Input: C, kernel, kernel parameters, epsilon
- Initialize b and all α's to 0
- Repeat until KKT satisfied (to within epsilon):
  - Find an example *e1* that violates KKT (prefer unbound examples here, choose randomly among those)
  - Choose a second example *e2*. Prefer one to maximize step size (in practice, faster to just maximize $|E_1 - E_2|$). If that fails to result in change, randomly choose unbound example. If that fails, randomly choose example. If that fails, re-choose e1.
  - Update α_1 and α_2 in one step
  - Compute new threshold b

At End of Each Step, Need to Update *b*

- Compute b_1 and b_2 as follows

$$b_1 = E_1 + y_1(\alpha_1^{\text{new}} - \alpha_1)K(\vec{x}_1, \vec{x}_1) + y_2(\alpha_2^{\text{new,clipped}} - \alpha_2)K(\vec{x}_1, \vec{x}_2) + b$$

$$b_2 = E_2 + y_1(\alpha_1^{\text{new}} - \alpha_1)K(\vec{x}_1, \vec{x}_2) + y_2(\alpha_2^{\text{new,clipped}} - \alpha_2)K(\vec{x}_2, \vec{x}_2) + b$$

- Choose $b = (b_1 + b_2)/2$
- Once KKT conditions are satisfied, if α_1 and α_2 were not clipped, it is in fact guaranteed that $b = b_1 = b_2$

Appendix

Proof that the update rule optimizes the functions with respect to the two selected examples, to the extent possible under the constraint both weights must stay between 0 and C

Derivation of SMO Update Rule

- Recall objective function we want to optimize:

$$\min_{\vec{\alpha}} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j K(\vec{x}_i,\vec{x}_j)\alpha_i\alpha_j - \sum_{i=1}^{N}\alpha_i,$$
$$0 \le \alpha_i \le C, \forall i,$$
$$\sum_{i=1}^{N} y_i\alpha_i = 0.$$

- Can write objective as:

$$\Psi = \frac{1}{2}K_{11}\alpha_1^2 + \frac{1}{2}K_{22}\alpha_2^2 + sK_{12}\alpha_1\alpha_2 + y_1\alpha_1 v_1 + y_2\alpha_2 v_2 - \alpha_1 - \alpha_2 + \Psi_{\text{constant}}$$

α_1 with everybody else

α_2 with everybody else

all the other α's

where: $K_{ij} = K(\vec{x}_i,\vec{x}_j),$

$$v_i = \sum_{j=3}^{N} y_j\alpha_j^* K_{ij} = u_i + b^* - y_1\alpha_1^* K_{1i} - y_2\alpha_2^* K_{2i}$$

Starred values are from end of previous round

Expressing Objective in Terms of One Lagrange Multiplier Only

- Because of the constraint $\sum_{i=1}^{N} y_i \alpha_i = 0$, each update has to obey:

$$\alpha_1 + s\alpha_2 = \alpha_1^* + s\alpha_2^* = w$$

where *w* is a constant and *s* is $y_1 y_2$

- This lets us express the objective function in terms of α_2 alone:

$$\Psi = \tfrac{1}{2} K_{11}(w - s\alpha_2)^2 + \tfrac{1}{2} K_{22}\alpha_2^2 + sK_{12}(w - s\alpha_2)\alpha_2$$
$$+ y_1(w - s\alpha_2)v_1 - w + s\alpha_2 + y_2\alpha_2 v_2 - \alpha_2 + \Psi_{\text{constant}}$$

Find Minimum of this Objective

- Find extremum by first derivative wrt α_2:

$$\frac{d\Psi}{d\alpha_2} = -sK_{11}(w - s\alpha_2) + K_{22}\alpha_2 - K_{12}\alpha_2 + sK_{12}(w - s\alpha_2) - y_2 v_1 + s + y_2 v_2 - 1 = 0$$

- If second derivative positive (usual case), then the minimum is where (by simple Algebra, remembering *ss* = 1):

$$\alpha_2(K_{11} + K_{22} - 2K_{12}) = s(K_{11} - K_{12})w + y_2(v_1 - v_2) + 1 - s$$

Finishing Up

- Recall that w = $\alpha_1 + s\alpha_2 = \alpha_1^* + s\alpha_2^*$ and

$$v_i = \sum_{j=3}^{N} y_j \alpha_j^* K_{ij} = u_i + b^* - y_1 \alpha_1^* K_{1i} - y_2 \alpha_2^* K_{2i}$$

- So we can rewrite

$$\alpha_2(K_{11} + K_{22} - 2K_{12}) = s(K_{11} - K_{12})w + y_2(v_1 - v_2) + 1 - s$$

as: $\alpha_2(K_{11} + K_{22} - 2K_{12})$ = $s(K_{11}-K_{12})(s\alpha_2^*+\alpha_1^*)$ +
$y_2(u_1-u_2+y_1\alpha_1^*(K_{12}-K_{11}) + y_2\alpha_2^*(K_{22}-K_{21})) + y_2y_2 - y_1y_2$

- We can rewrite this as: $\alpha_2(K_{11} + K_{22} - 2K_{12}) =$

$$\alpha_2^*(K_{11} + K_{22} - 2K_{12}) + y_2(u_1 - u_2 + y_2 - y_1)$$

Finishing Up

- Based on values of *w* and *v*, we rewrote

$$\alpha_2(K_{11}+K_{22}-2K_{12}) = s(K_{11}-K_{12})w + y_2(v_1-v_2)+1-s$$

as:

$$\alpha_2(K_{11}+K_{22}-2K_{12}) =$$

$$\alpha_2^*(K_{11}+K_{22}-2K_{12}) + y_2(u_1-u_2+y_2-y_1)$$

- From this, get update rule: