CoCoA @ Amazon

Simone Forte
ETH Zurich
fortesi@student.ethz.ch

1 CoCoA for Logistic Regression

The initial solver has been restructured to be more general and allow different loss functions, local solvers and losses. The main structure of the algorithm stays pretty much the same but there now exist the possibility to specify what local solver to use, allowing to choose also fully primal solvers; the only local solver currently implemented is SDCA.

1.1 SDCA

The SDCA algorithm is the same as it was in the original CoCoA repository but an abstraction has been created to allow for custom "Single coordinate optimizers" (SCO). What the SCO does is, given a point (x, y), the old α value and the old w vector, is solving the following problem:

$$-\frac{\lambda n}{2}\left\|\boldsymbol{w}^{(h-1)}+\frac{1}{\lambda n}\Delta\alpha\,\boldsymbol{x}_i\right\|^2-\ell_i^*\big(-(\alpha_i^{(h-1)}+\Delta\alpha)\big)$$

and returning a better value for $\Delta\alpha$ (non necessarily the optimum).
It is also possible, and implemented in the library, to solve the primal SCO directly, optimizing on $\boldsymbol{w}_i$:

$$\frac{1}{n}\ell_i((\overline{\boldsymbol{w}}+\boldsymbol{w}_i)^T\boldsymbol{x}_i)+\frac{\lambda}{2}\left\|\boldsymbol{w}_i\right\|^2.$$

This problem can be obtained by looking at the local primal problem as defined in the CoCoA paper and assume that we have two partitions where one is made by point $\boldsymbol{x}_i$ only and the other by all the other points. We can also rewrite it as:

$$\frac{1}{n}\ell_i((\boldsymbol{w}^{(h-1)}-\boldsymbol{w}_i^{(h-1)}+\boldsymbol{w}_i)^T\boldsymbol{x}_i)+\frac{\lambda}{2}\left\|\boldsymbol{w}_i\right\|^2$$

But since $\boldsymbol{w}_i^{(h-1)}$ and $\boldsymbol{w}_i$ need to be a linear combination of the vectors in the partition, they thus have to be a scaling of $\boldsymbol{x}_i$, respectively $\alpha^{(h-1)}\boldsymbol{x}_i$ and $\alpha\boldsymbol{x}_i$, giving us:

$$\frac{1}{n}\ell_i\big((\boldsymbol{w}^{(h-1)})^T\boldsymbol{x}_i-\alpha^{(h-1)}\left\|\boldsymbol{x}_i\right\|^2/(\lambda n)+\alpha\left\|\boldsymbol{x}_i\right\|^2/(\lambda n)\big)+\frac{\lambda}{2}\left\|(\alpha^{(h-1)}+\Delta\alpha)\boldsymbol{x}_i/(\lambda n)\right\|^2$$

finally getting

$$\ell_i\Big(\boldsymbol{x}_i^T\boldsymbol{w}+\Delta\alpha\left\|x\right\|^2/(\lambda n)\Big)+\frac{(\alpha^{(h-1)}+\Delta\alpha)^2}{\lambda n}\left\|\boldsymbol{x}_i\right\|^2.$$

Solving this problem (with respect to $\Delta\alpha$) has several advantages over solving the dual form:

- It allows to solve the problem without the need of a pen-and-paper derivation of the convex conjugate.

- It requires no explicit constraint over the domain of the α variables, which is instead needed for the the dual form. This allows to use solvers such as SGD or Newton's method, instead of requiring more complex bracketing based methods such as Brent's method.

1.2 Logistic regression

The logistic regression problem can be solved using different methods for either the primal or the dual single-coordinate problem:

- BrentOptimizer for the dual SC problem: it uses Brent's method from the apache-commons library to minimize a custom loss function; in our case the dual SC problem. The method does not require any derivative being plugged in. The exact form of the convex conjugate loss used is:

  $$\ell_y^*(\alpha) = (1 + \alpha y) \log (1 + \alpha y) - \alpha y \log (-\alpha y)$$

  having domain equal to $(-1, 0)$ for $y = 1$ and $(0, 1)$ for $y = -1$.

- BrentOptimizer for the primal SC problem: same as before but it now solves the primal SC problem. The loss is of course:

  $$\ell_y(p) = \log (1 + \exp (-py)).$$

- Newton's method for the primal SC problem: being the primal SCP unconstrained we can use Newton's method, feeding to it the first and the second derivative of the primal SCP. This two are constructed using the first and second derivative of ℓ_y:

  $$\ell'_y(p) = \frac{-y}{1 + \exp (py)}$$

  and

  $$\ell''_y(p) = \frac{y \exp(py)}{(1 + \exp (py))^2}.$$

  The derivatives of $\ell_i\Big(\boldsymbol{x}_i^T \boldsymbol{w} + \Delta\alpha \left\|x\right\|^2 / (\lambda n)\Big)$ with respect to $\Delta\alpha$, needed to construct the primal SCP to be fed to Newton's, are then computed in the code using the chain rule. This allows for a cleaner and simpler definition of new loss functions (and their derivatives).